NASA Contractor Report 187196

Probabilistic Structures Analysis Methods for Select Space Propulsion System Components (PSAM)

(2nd Annual Report)

Southwest Research Institute
San Antonio, Texas

September 1991

Prepared for
Lewis Research Center
Under Contracts NAS3-24389

NASA
National Aeronautics and
Space Administration

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

1.0 PROGRAM PLAN OUTLINE AND NARRATIVE

1.1 Introduction

This Annual Report focuses on the effort that has been completed during the second year of the technical effort. The total project is now expected to last a total of five years. All elements of the technical tasks to be accomplished have now been defined. The new effort in the third year is the initiation of programming of the advanced methods formulation; the approximate methods effort will not begin until the fourth year of technical effort.

The project is very much a team effort with significant contributions coming from several task managers: Dr. O.H. Burnside - Verification/Validation Coordination; Dr. Y.-T. Wu - NESSUS/FPI Development; Drs. J. Nagtegaal, S. Nakazawa and Mr. J. Diaz - PFEM Development; Dr. K.R. Rajagopal - Verification Studies; Dr. P. Fink - NESSUS/EXPERT Development; Prof. P. Wirsching - Advanced Simulation Methods; and Prof. S. Atluri - Hybrid FEM Development and Level III Modeling. The SwRI Program Manager acknowledges the critical contributions from each of these individuals.

The remainder of this Section outlines the elements of the technical approach being taken in PSAM. Section 2.0 summarizes the technical accomplishments of the second year of the project, supported by various appendices. Section 3.0 presents a brief outline of some of the current efforts.

1.2 Probabilistic Finite Element Methods (PFEM) Plan

The developed methods of analysis are to treat linear problems as well as those with nonlinear material and geometric response. Stochastic modeling of loads (e.g., centrifugal, thermal, pressure), geometry, and material behavior are being modeled with three levels of approximation, relative to accuracy and confidence. Level I analyses treat randomness as being spatially homogeneous (e.g., each part has a different modulus, yield stress, thermal load, etc.). Level II analyses treat random variables as random fields (e.g., modulus variability is different in the bore of a disk, versus the rim of the disk; pressure uncertainty is different at the root of an airfoil versus the tip of the blade). Level III stochastic modeling is to be able to reflect uncertainty between variables in the governing equations (e.g., strain agrees with displacement gradients only

in stochastic, not deterministic, terms; stress is related to strain through stochastic relations).

Two methods of probabilistic modeling are included in the various analysis methods. The first of these is the Fast Probability Integration (FPI) method. The FPI method is adopted from the field of structural reliability as a way of predicting the probability that a response variable (e.g., stress, frequency) will exceed some allowable. The method is based on establishing the approximate sensitivity of a response variable to the stochastic variables, and then processing these sensitivities by the FPI algorithm to establish the cumulative response distributions for the variables. The second method is direct simulation using an enhanced Monte Carlo method. Both probabilistic prediction methods will make use of the same structural sensitivity data base, which is generated by NESSUS. Confidence levels will be estimated for the response variable distributions that are calculated. A composite load spectrum analysis procedure will be included.

The PFEM is a direct adaptation of standard finite element methodology to the needs of PSAM. The finite element code (NESSUS/FEM) is to include plate and shell elements based initially on the displacement method of formulation, and on linear equations of motion and material behavior. Hybrid plate and shell elements are to be included, as well as nonlinear geometric and material behavior. The NESSUS/FEM code will include a variety of standard finite elements for structural modeling. The NESSUS/FEM program will allow for nonlinear elastoplastic/creep modeling, and for geometric nonlinear problems of finite displacement, rotation, and strain.

In addition, an enhanced shell/plate element formulation will be developed. This enhanced formulation will be a quasi-continuum element that provides for surface data input and nodal stress recovery, consistent with the requirements of the NASA SOW. The enhanced element is a displacement formulation, developed from the Hu-Washizu variational formulation. Stresses, strains, and displacements will be interpolated independently. In order to reduce the formulation to a displacement-like formulation, the stress and strain fields are discontinuous between the elements. Displacements will be interpolated on a nodal basis, with nodes selected at the surfaces of the shell/plate element. The element will satisfy all constant stress modes and will provide full rigid body mode

capability (i.e., has correct rank). The eight noded element will provide for surface pressure load definition, as well as for nodal stress, strain recovery.

The hybrid element formulation is also based on the Hu-Washizu variational statement. Thus, it will have stress modes that are defined independent of the displacement modes. The element will be a sixteen node shell/plate element with surface loading and nodal stress, strain recovery capability. Special interpolation capability for severe thermal gradients is planned in both the enhanced and hybrid shell/plate formulations.

Material response is to include the range from elastic to thermoviscoplastic. The material model will be based on theoretical development for a random relationship between stress and strain for the general class of thermomechanical response problems. The material modeling considerations will allow for a full, Level III interpretation of stress/strain stochasticity. The model will be based on the assumption that each material has its own stochastic response over the full range of loading history. Thus, we rule out as a mathematical construct, the notion of incremental stochasticity. The theoretical material modeling development will admit implementation of endochronic or thermoviscoplastic considerations.

The NESSUS code is modular for adaptation to the General Purpose Structural Analysis (GPSA) framework. The modules include NESSUS/FEM, NESSUS/PAAM, NESSUS/BEM, NESSUS/FPI, NESSUS/PRE, NESSUS/EXPERT, and others as needed. Interfaces between these modules will be clearly defined.

The approach to validation is to perform validation and verification studies on the new element and formulation capabilities as they become available. This will also provide for direct comparisons between the various solution capabilities. The NESSUS code is to be continually validated through its application to well-defined problems with known probabilistic responses in order to demonstrate the full and reliable capability of the code.

It has been found to be very important that the verification study include a wide range of simple structural models that exercise the various options of the NESSUS code. These verification problems, being run by SwRI and Rocketdyne, serve to provide further confidence in the code, to develop rule bases for NESSUS/EXPERT, and demonstrate the utility of the code.

The NESSUS code will be verified by its application to four selected space propulsion system hardware items. These will include the turbine blade, transfer duct, LOX post, and the high pressure oxidizer duct. Experimental data to support the analyses will be compiled and statistically modeled. The four verification problems are outlined in Appendix A.

1.3 Probabilistic Approximate Analysis Methods (PAAM) Plan

The PAAM code will be established by SwRI in consultation with Rocketdyne staff. The purpose of the PAAM code is to provide a mechanics of materials approach to the probabilistic modeling of plate and shell type structures. The approach to be taken by SwRI will be to:

1. Identify simplified plate/shell problems representative of plate and shell regions within the four selected propulsion system components.

2. Identify plate and shell type analytical solutions that correspond best to the physical problems identified in 1., above.

3. Modify the analytical solutions to account, in a suitable and approximate manner, the loading, material response, and structural response features required for the four component problems.

4. Program NESSUS/PAAM to include a library of these solutions and approximation methods for loading, material response, and structural response.

1.4 Probabilistic Advanced Methods (PAdvAM) Plan

The basis of the Probabilistic Advanced Analysis Methods is the boundary element method, specifically the BEST3D code previously developed under NASA HOST funding. SwRI has further developed this code and proposes to modify it in a manner suitable for inclusion in the PSAM analysis library as NESSUS/BEM.

The boundary element method (BEM) contrasts, for the linear problem, with the finite element method (FEM) by the fact that the governing equations are written at the boundary of the body only. The so-called boundary integral equation (BIE) governs the relationship between tractions and displacements at the surface of the body. The only geometric description of the body that is required is the surface of the body. For the thermoelastic problem with variable material properties and problems of linear vibration, it is also possible to reduce the continuum problem to a boundary formulation. For problems with geometric or material nonlinearities, and for transient dynamic problems, a volume modeling is generally required.

The perturbation algorithm will be developed for NESSUS/BEM. For those problems with no volume integrations, all perturbations will be in terms of surface data. Geometry, for example, will be perturbed through explicit differentiation of the boundary modeling shape functions. The perturbations will maintain continuity of boundary shape by moving the boundary node locations. Perturbations of the mass matrix for vibration analysis will be similarly modeled. Level I material perturbations will be explicitly accounted for.

Level II and Level III material perturbations will be examined using one of two possible approaches. The most direct is to handle these through volume integrals (discussed below). The most interesting is to develop boundary models that can interpolate volumetric changes, in terms of perturbed boundary data. The latter approach is favored and will be the first to be explored. Explicit differentiation or differencing of boundary data will be used in order to avoid an iterative solution algorithm.

Volume integration methods will be especially developed for NESSUS/BEM to take advantage of plate/shell type of behavior. Simplifying interpolation assumptions will be made to reduce the need for significant numbers of volume discretizations in the through-thickness direction. It will be assumed that deviations in strain behavior in this direction from the linear solution are not excessive.

The first year (FY87) will establish the linear static and dynamic thermoelastic modeling capability for NESSUS/BEM. The second year (FY88) will focus on the establishment of the essential nonlinear modeling capability, but without the full Level III thermoviscoplastic modeling and random transient loading. The third and final year (FY89) will release the full nonlinear capability.

The stochastic basis of a variational model of structural response will be established by GIT researchers under the direction of Professor Satya Atluri. The stochastic variational statement will be used to demonstrate the formulation basis of the Level I, II NESSUS PFEM models. Further, it will be used to establish the Level III formulation for adoption into NESSUS. It is expected that the Level III model will be based on the use of correlation model matrices linking the strains and the displacement gradients, and another linking stress to the material constitutive behavior.

2.0 TECHNICAL PROGRESS SUMMARY

2.1 Task I: PFEM

2.1.1 NESSUS/FEM Development

2.1.1.1 Status at End FY85

The finite element analysis module NESSUS/FEM has evolved from the MHOST code, developed by MARC for Pratt and Whitney Aircraft Company under NASA contract NAS3-23697. A review of the capabilities of MHOST by SwRI indicated the need for enhancements to provide additional features relevant to the analysis of reusable space propulsion system components. This enhanced version of the MHOST code was delivered to SwRI in August 1985 as NESSUS 0.1, and was the latest version of the code shipped from MARC prior to the end of FY85.

The major enhancements provided with NESSUS 0.1 included:

A. Element library and problem modeling features
   - o Addition of a two-noded Timoshenko beam element
   - o Rotational inertia terms in consistent mass matrices
   - o Grounded springs of prescribed stiffness
   - o More convenient definition of time-histories for pulse loading

B. Algorithmic enhancements
   - o Displacement method option for linear elastostatics
   - o Power shift option for eigenvalue extraction

C. Analysis capabilities for linear systems
   - o Transient dynamics using mode superposition
   - o Harmonic loading and base excitation
   - o Random vibration (PSD) analysis

By the end of FY85, the basic formulation for probabilistic finite element analysis as implemented in NESSUS had been developed and demonstrated on a few sample problems. The original approach relied on a Taylor series expansion of the stochastic problem about a deterministic solution. This approach did not appear to be practical for the large systems of finite element equations parameterized by many random variables that are needed for realistic SSME applications. An alternative approach was developed, based on an iterative perturbation analysis method that uses the factorized stiffness of the unperturbed system as the iteration preconditioner for obtaining the solution to the perturbed problem. This approach eliminates the need to compute, store and manipulate explicit partial derivatives of the element matrices and force vector, which not only reduces

memory usage considerably, but also greatly simplifies the coding and validation tasks. A similar approach for the solution of the perturbed symmetric eigenproblem was developed by Professor Juan Simo, at the Applied Mechanics Division, Stanford University, for implementation in NESSUS/FEM.

The efficient treatment of correlated random variable fields was identified early on in the PSAM effort as a major practical issue, since many SSME applications involve random variables that are correlated to some degree. Examples of this include random pressure and temperature fields defined on the surface of a turbine blade, or the thickness of the walls and liners in the transfer ducts or nozzle of a rocket engine. The strategy adopted in the NESSUS code relies on a variable transformation into the eigencoordinates of the covariance matrix defining the random field. The transformed variables can be shown to be uncorrelated and may therefore be manipulated as such in the NESSUS/FEM and NESSUS/FPI modules. The computation of the transformed variables may be carried out prior to the finite element analysis of the model and may, therefore, be regarded as a pre-processing operation.

Several aspects of the proposed formulation were demonstrated on an ad-hoc basis before the end of FY85. The feasibility of the iterative perturbation algorithm for elastostatics was demonstrated in April 1985 with a problem involving a clamped square plate under uniform pressure loading, using a 10 x 10 mesh of shell elements and subjected to thickness variations along one edge. The numerical manipulations proposed for handling correlated data were demonstrated also in April 1985 with a problem involving a scalar random variable field defined on a 10 x 10 grid with varying strength of correlation. Finally, all ingredients for the proposed formulation were combined in a demonstration problem using a simplified model of a curved turbine blade discretized with 48 shell elements, and having random pressure and temperature fields with partial correlation, random uniform thickness, and random stiffness at the root. This exercise was completed in May 1985. Although the formulation for the iterative solution of the perturbed symmetric eigenproblem was essentially complete by the end of FY85, no demonstration problems using this approach were available at the time.

2.1.1.2 Database Development

The perturbation database (Fig. 2.1) provides an external record of the perturbation data obtained during execution of the NESSUS/FEM module. In a typical NESSUS/FEM execution, a number of perturbed solutions about a deterministic state are computed with the use of appropriate numerical algorithms. The results for both the unperturbed and all perturbed systems are added to the perturbation database as soon as each converged solution becomes available. The information stored in the database may then be accessed by the NESSUS/FPI module, in order to extract the data required for the computation of a system reliability estimate or to obtain distribution curves for relevant response variables. The perturbation database is problem-specific, and was designed to centralize all the information pertinent to the analysis of a given model, even if it is obtained in the course of multiple NESSUS/FEM executions. Future releases of NESSUS/FEM will allow full use of these capabilities.

The perturbation database resides in a binary (unformatted) direct-access file, and may be accessed using standard FORTRAN I/O facilities. The database is structured as a two-way ordered linked list (Fig. 2.2), allowing for quick and efficient traversal in search of a specific entry. This type of data structure allows for the insertion and deletion of individual entries anywhere in the list without violating the original ordering convention (Fig. 2.3). It is therefore possible to enrich the existing database with information obtained in multiple executions of NESSUS/FEM without having to regenerate data obtained in previous runs.

The organization of a typical database constructed by NESSUS/FEM is outlined in Fig. 2.4. The entry point is a single PROBLEM HEADER RECORD, always occupying the first physical record in the file. This record contains sizing information pertinent to the problem, together with pointers to two distinct ordered linked lists. One list contains the load incrementation history, with the individual perturbation data sets nested inside each increment. The second list contains the eigenvalue and eigenvector data, this time with the individual eigenpairs nested inside each perturbation data set. Both lists consist in a series of INCREMENTAL or EIGENPAIR DATA HEADER RECORDS, forming two two-way ordered linked lists, shown in Fig. 2.4 as extending downward and upward from the single entry point. These headers in turn contain pointers to the actual DATA RECORDS, containing

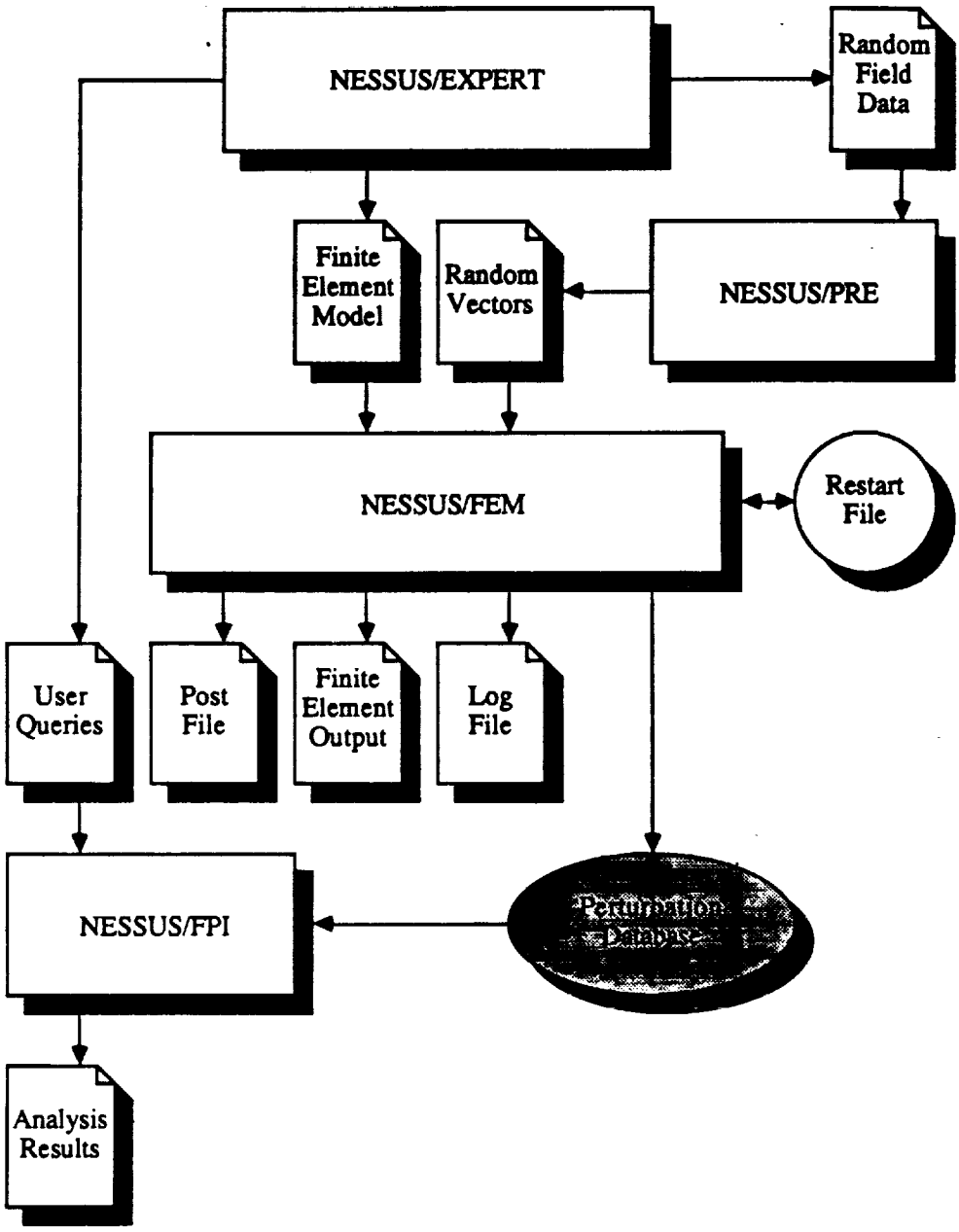


Fig. 2.1 The NESSUS Perturbation Database

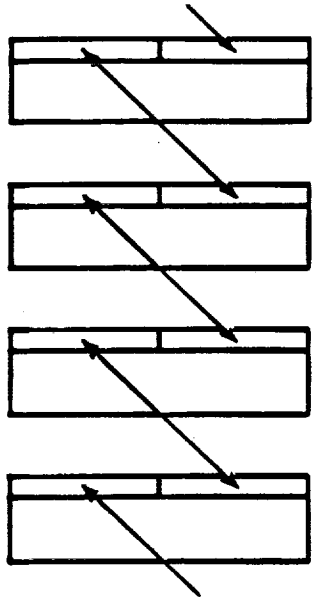


Fig. 2.2 A Two-way Linked List

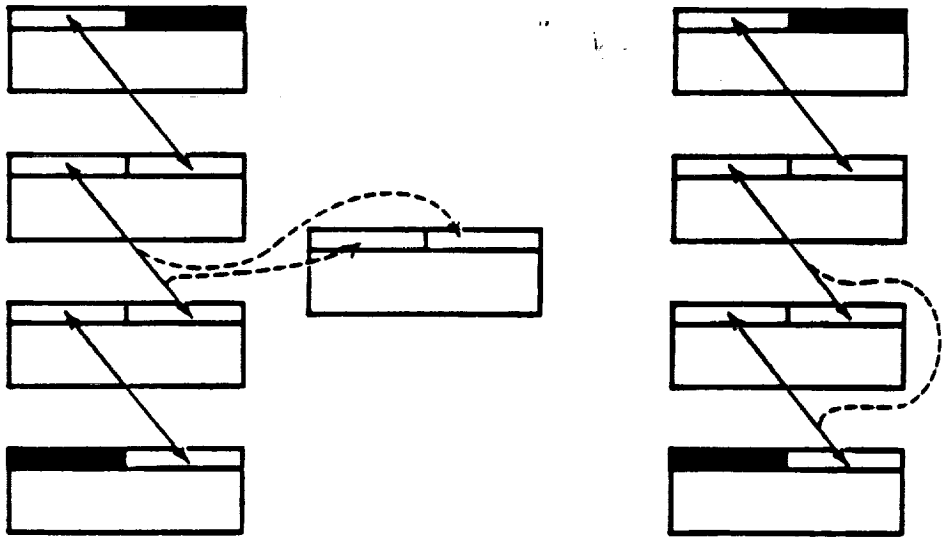


Fig. 2.3 Insertion of a New Item in a Two-way Linked List, and Deletion of Another Item from a Similar Linked List, as Shown by the Dashed Lines

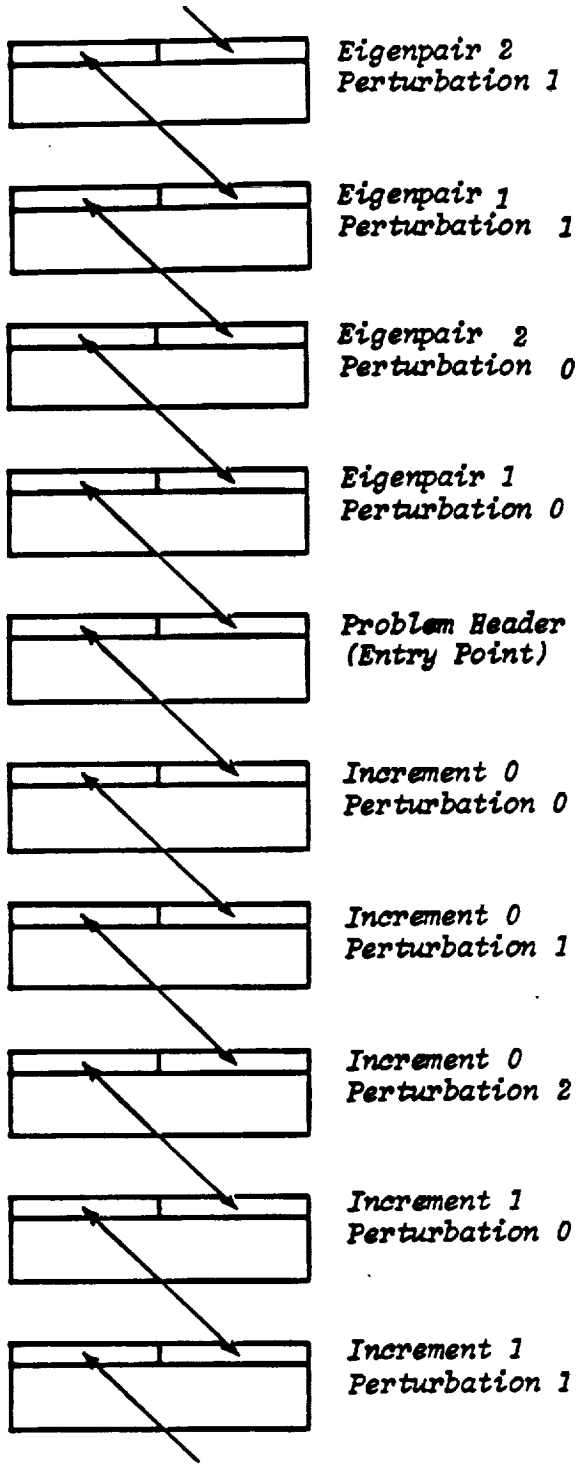


Fig. 2.4 Data Structure of the NESSUS Perturbation Database

information on the type of perturbation and the perturbed system response. A null pointer is used to flag the unavailability of data, which may result from the lack of a converged solution. Null pointers are also used to terminate both the incremental and eigenpair data lists.

The present implementation of the perturbation database provides easy and efficient data retrieval using standard algorithms for manipulating ordered linked lists. Insertion and deletion of individual entries can be accomplished locally without the need for moving large blocks of data. The internal data structure was designed with the flexibility to accommodate additional capabilities planned for future releases of NESSUS/FEM with minimal adjustments to the software already in place. The use of binary (unformatted) files provides compact storage for the potentially massive amounts of data required for the analysis of realistic problems. For small problems, a simple FORTRAN utility is available to provide translation of the database into formatted (printable) form. This can be quite useful for debugging codes written to access the database, or for moving small databases across different computer systems. The internal data structure of the perturbation database is well documented in a report which can be used as a guide for the development of new codes requiring access to existing databases. Finally, it should be noted that the information contained in the perturbation database may be useful for applications other than probabilistic structural analysis, such as the investigation of the sensitivity of the response to several design parameters.

2.1.1.3 NESSUS/PRE Module Development

The NESSUS/PRE module (Fig. 2.5) is a pre-processor used for the preparation of the statistical data needed to perform probabilistic finite element analysis with NESSUS/FEM. NESSUS/PRE allows the user to describe a spatial domain defined by a set of discrete points, typically corresponding to the nodal points of an existing finite element mesh. One or more random variable fields may then be specified over this spatial domain by defining the mean value and standard deviation of the field variables at each at each point, together with the appropriate form of correlation. Each random variable field may be modeled as uncorrelated, fully correlated or partially correlated. The current version of NESSUS/PRE limits the treatment of partially correlated fields to fields of Gaussian variables with equal correlation strength in all directions (isotropic correlation).

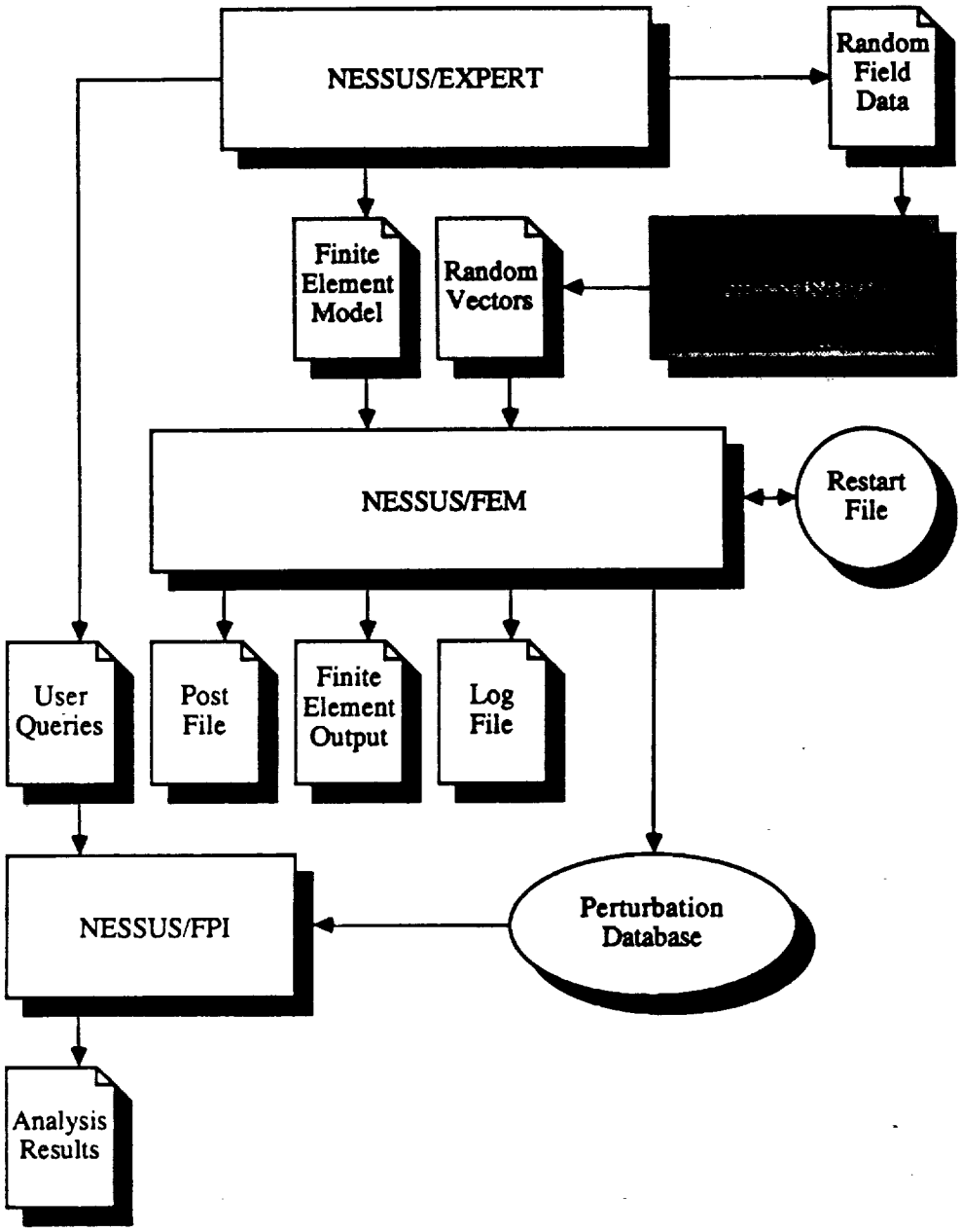


Fig. 2.5 The NESSUS/PRE Module

Each random field is treated according to the form of correlation specified for it. Uncorrelated random fields are automatically decomposed into a set of uncorrelated vectors, each corresponding to a unit variation at a given degree-of-freedom. Fully correlated fields are automatically converted to a single vector, corresponding to a scaling of the random field by one standard deviation. For a partially correlated random field, the preprocessing operation in NESSUS/PRE is considerably more complex. This will involve the construction of the variance-covariance matrix for the field, followed by the spectral decomposition of this matrix. The field data is then transformed into the eigencoordinates of the covariance matrix, yielding a set of mutually uncorrelated random vectors which contain all the information present in the original correlated field. The theoretical details of this procedure are given in Section 5.3.1 of the PSAM First Annual Report. The spectral decomposition of the covariance matrix is performed conservatively, using Jacobian iteration to solve simultaneously for all eigenvalues and eigenvectors of the matrix. If the correlation is strong, the uncertainty in the data is dominated by just a few of the highest eigenvalues of the matrix. Hence, the user is given the option to simplify the problem by truncating the spectrum to a prescribed tolerance, retaining only the most significant eigenvalues for the analysis. This strategy can produce a very significant reduction in the amount of computation required for the analysis, especially in problems involving a large number of random variables. In all cases, the output from NESSUS/PRE will consist of a set of uncorrelated random vectors written to an external formatted data file. This file will contain the random variable definitions for NESSUS/FEM, and may be included in the input deck to the finite element module without further modification.

The present implementation of NESSUS/PRE allows the specification of random fields involving:

1. nodal coordinate data
2. nodal shell thickness
3. nodal shell or beam normals
4. thickness of plane stress elements
5. modulus of elasticity
6. Poisson's ratio

7. thermal expansion coefficient
8. material density
9. rotational speed
10. nodal force vectors
11. element pressures and edge tractions
12. nodal temperatures
13. elastic beam section properties
14. base spring stiffnesses
15. orientation of anisotropy axes

Additional types of random variables will be included in future releases of NESSUS/PRE as required by the enhanced capabilities of NESSUS/FEM.

2.1.1.4 Code Structure

The NESSUS/FEM module (Fig. 2.6) provides finite element modeling and analysis capabilities for probabilistic structural analysis problems. The finite element code is structured as a set of six major driver routines, reflecting the types of analysis currently available. These include:

1. A static analysis driver for the solution of linear and nonlinear problems in either a purely iterative manner or in incremental-iterative fashion.
2. A bifurcation buckling driver, used for stability analysis of linearized structural systems.
3. A modal extraction driver for the determination of the undamped natural frequencies and mode shapes for vibrating structures.
4. A mode superposition driver for the analysis of steady state or transient linear vibration problems in the time domain.
5. A random vibration driver for the analysis of problems involving stationary random excitation by integration in the frequency domain.
6. A direct time integration driver using the Newmark-b method for the solution of linear and nonlinear transient dynamics problems.

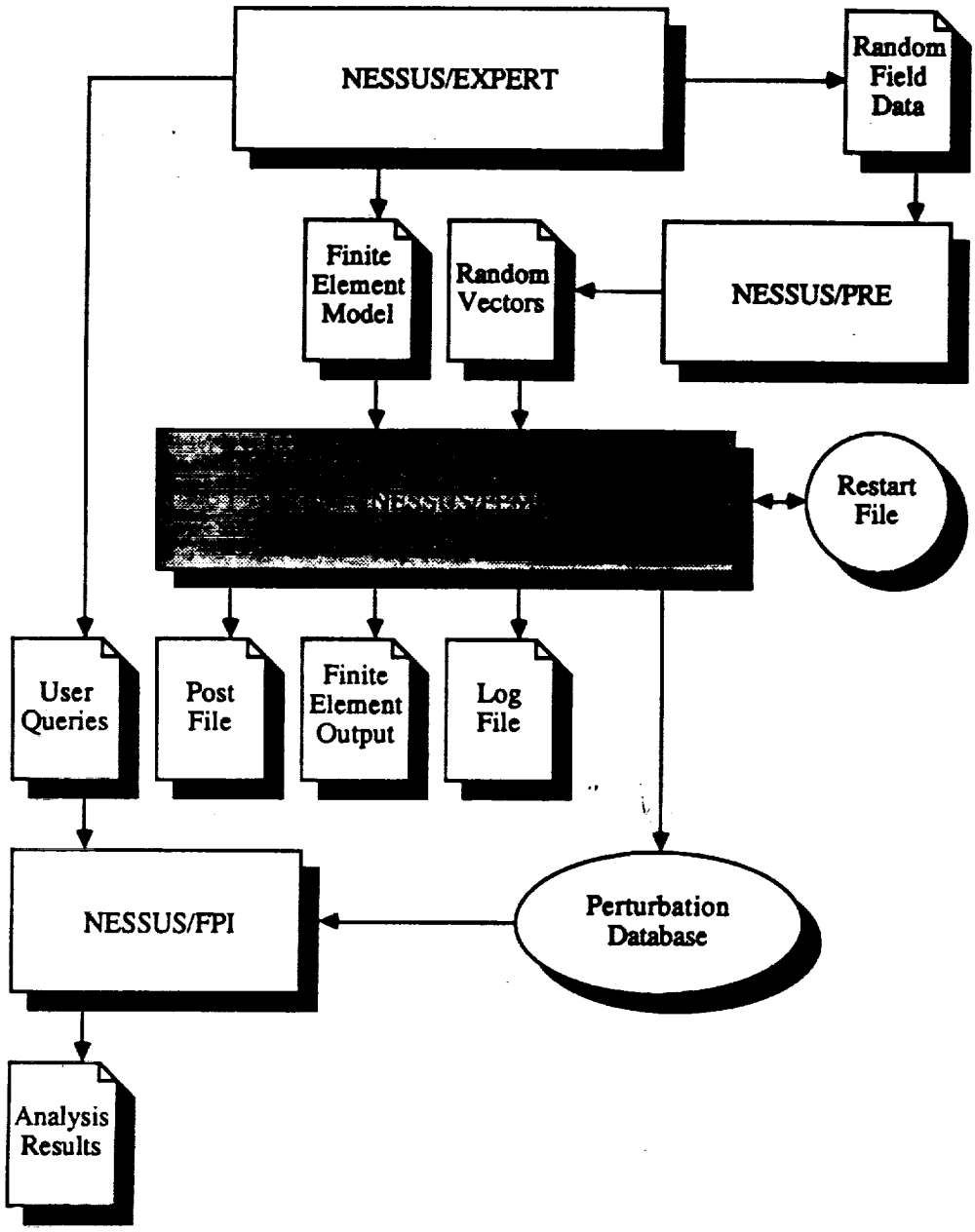


Fig. 2.6 The NESSUS/FEM Module

A new facility has been added at the topmost level of NESSUS/FEM to allow conditional transfer of control between driver routines. This feature allows the performance of more sophisticated types of analysis, which are useful for the realistic modeling of typical SSME components. A typical application might involve the static analysis of a spinning turbine blade, with centrifugal loading applied over a number of load increments. At a prescribed increment number, control of the execution may be transferred to the modal extraction driver, in order to determine the vibration characteristics of the blade, including centrifugal mass and stress stiffening effects due to the initial stresses obtained in the static analysis. These features were first available in Version 1.2 of the code.

Significant efficiency improvements were achieved by replacing the old band and frontal equation solvers with a newly developed profile solver. The new solver, available in Version 1.3, not only provided increased speed in the factorization and back substitution phases of the analysis, but also resulted in a substantial reduction of memory requirements for medium to large problems. This allowed the in-core solution of large turbine blade models using 8-noded bricks. Selected performance results for the new solver are summarized in Fig. 2.7 - Fig. 2.9. These numbers were obtained on the PRIME 9955 at MARC, with the memory requirements expressed in single precision (32 bit) words.

The extraction of eigenvalues for both linear dynamics and buckling problems is performed using the subspace iteration method. Multiple power shifts may be used to extract modes within prescribed frequency bounds. This technique is particularly useful in the analysis of structures containing rigid-body modes. The eigenvalue analysis subsystem is very similar to the one available in NESSUS 1.0, having been modified to accommodate profile storage for the stiffness and mass matrices, together with other minor efficiency improvements.

A full library of modern algorithms for nonlinear analysis is available in NESSUS/FEM. Both full Newton and modified Newton iteration algorithms have been available since Version 0.1. Newly implemented algorithms for nonlinear analysis include the line search algorithm, Davidon rank-one secant Newton update and inverse BFGS rank-two update. These algorithms have been available in Version 1.0 and up. Variations of

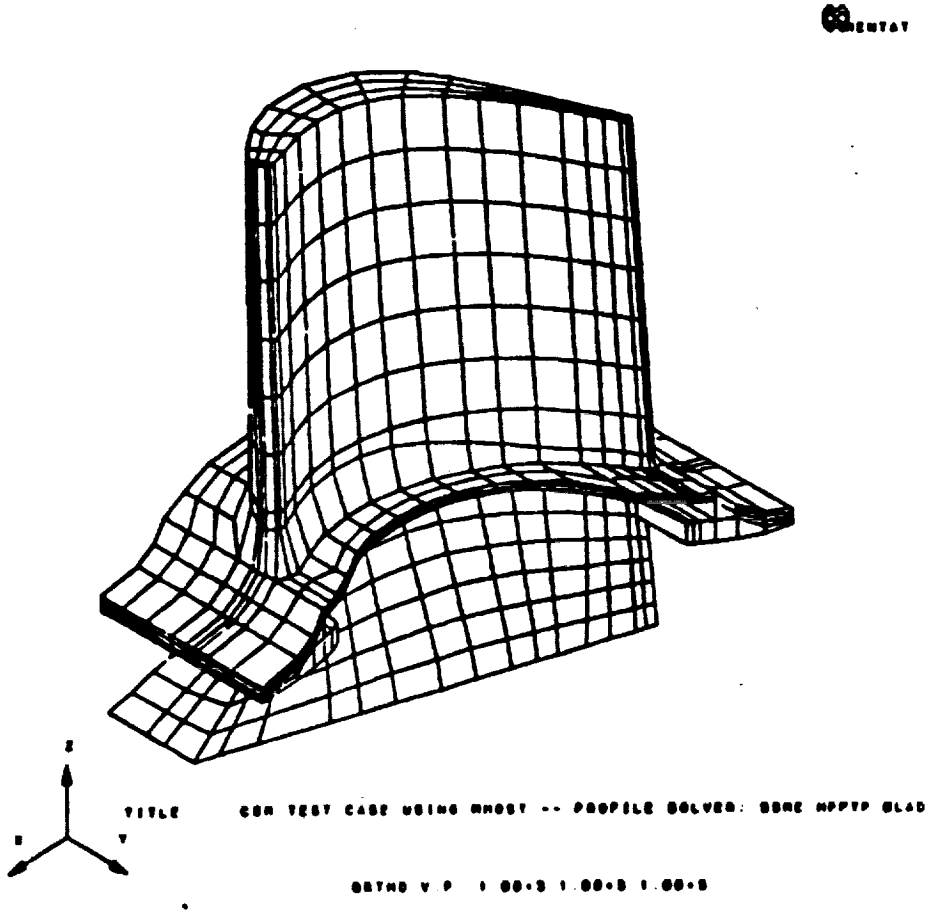


Fig. 2.7 SSME HPFTP Blade Model with 1025 Brick Elements and 1575 Nodes

	Profile Solver	Band Solver
Memory Requirement	3483811	4459647
Solution Time	1466.594 sec	N/A*

* Too Large to Run on PRIME 9955 at MARC

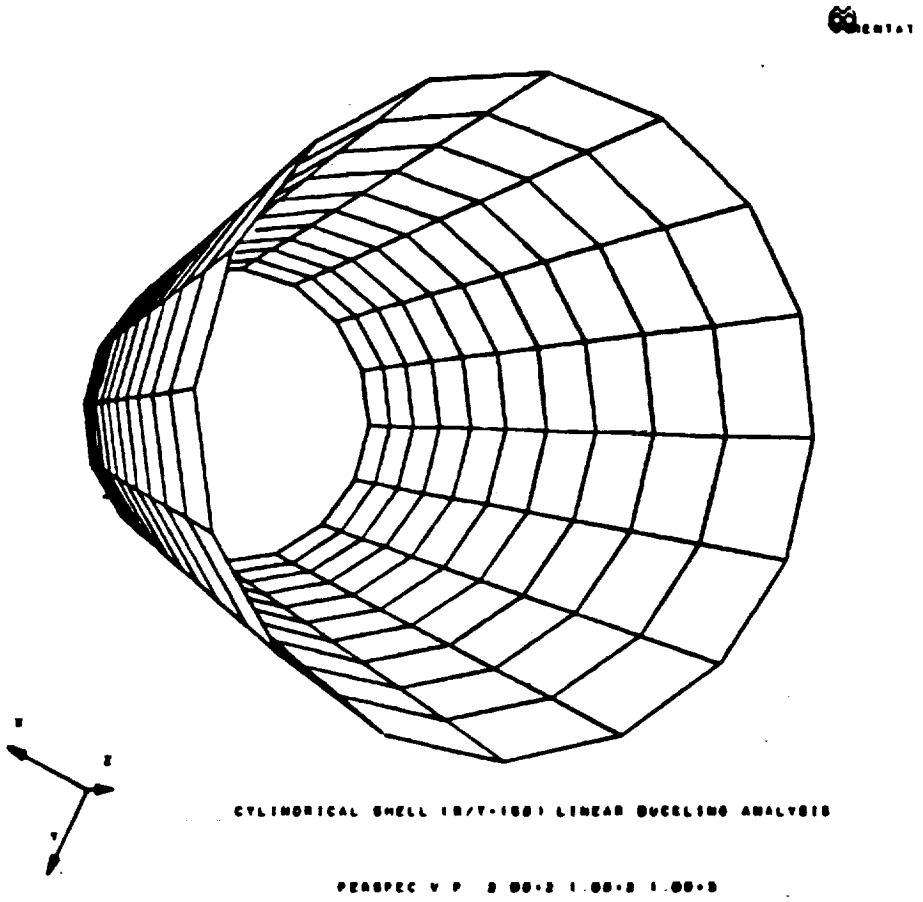


Fig. 2.8 Buckling Analysis of a Cylinder with 160 Shell Elements and 176 Nodes

	Profile Solver	Band Solver
Memory Requirement		
(a) Static	498873	596703
(b) Eigenvalue	1044773	1340375
Solution Time		
(a) Static	62.057 sec	171.430 sec
(b) Eigenvalue	66.788 sec	187.551 sec

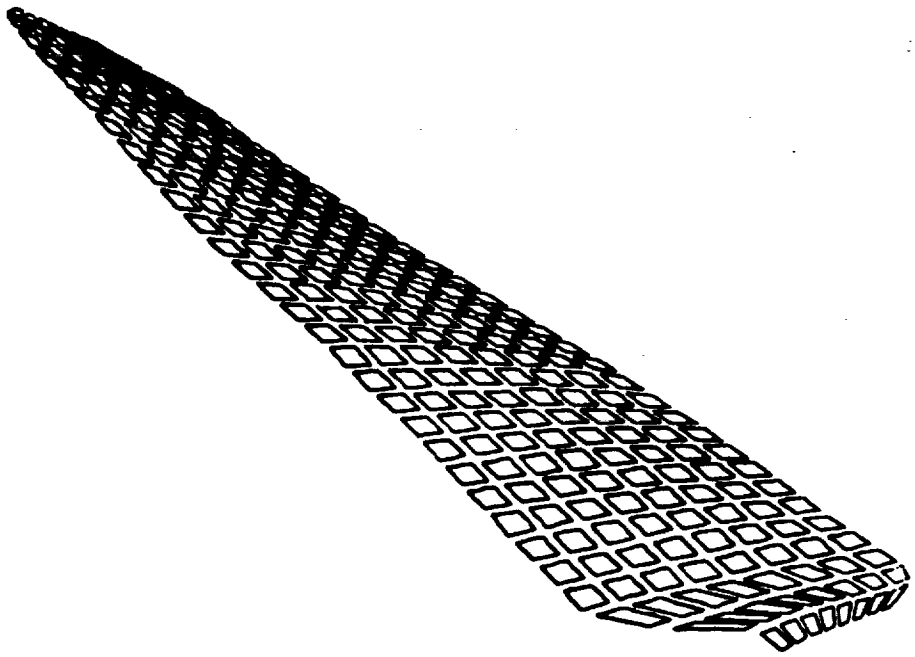


Fig. 2.9 Modal Analysis of a Composite Laminate Fan Blade with 240 Shell Elements and 279 Nodes

	Profile Solver	Band Solver
Memory Requirement	824341	1054259
Solution Time	24.794 sec	93.764 sec

many of these algorithms have since been applied to the computation of the solution to the perturbed elastostatic problem, as discussed in Section 2.1.1.6 of this report.

2.1.1.5 Element Technology

The element library currently available in NESSUS consists of six isoparametric, numerically integrated element types (see Table 2.1). Geometric quantities and material properties are defined at the nodes, and interpolated into the interior of each element using the appropriate shape functions. A nodal projection and smoothing algorithm is used to allow the reporting of strains and stresses on a nodal basis.

Continuum-type problems may be modeled using bilinear four-node quadrilaterals for plane stress, plane strain or axisymmetric situations, or trilinear eight-node bricks for three-dimensional problems. All B-matrix routines for continuum elements allow full, reduced, trapezoidal and selective integration. Selective integration is implemented using the B-bar approach, and has been designed to facilitate the implementation and testing of different integration weighting schemes. The performance of these elements has recently been improved with the adoption of a strain filtering scheme based on a local element orthogonal coordinate system constructed by polar decomposition of the Jacobian matrix for the isoparametric mapping. This technique enhanced the behavior of the element in situations involving distorted elements.

The shell element currently available in NESSUS is a four-node isoparametric formulation derived from the Reissner-Mindlin plate and shell theory. Bilinear interpolations are used for the coordinates, displacements and rotations. The element is selectively integrated, and stabilized by hourglass control on the transverse shear terms. An in-plane twist term is included to avoid the "drilling mode" singularity on a flat assembly of elements. This element may be used to model thick shell problems, with significant transverse shear deformation, and retains acceptable accuracy when used to model thin shell structures.

A two-node linear isoparametric beam element is also available, based on Timoshenko beam theory. Linear interpolations are used for the cross-section, displacements and rotations. Reduced one-point integration is used for economy, since this will yield a rank-sufficient stiffness matrix for the element. Since the cross-sectional properties for

Table 2.1

Summary of the NESSUS Element Library

	P.STRS	P.STRN	AXSYM	BRICK	SHELL	T. BEAM
ITYPE	3	11	10	7	75	98
NELCRD	2	2	2	3	3	6
NELNFR	2	2	2	3	6	6
NELNOD	4	4	4	8	4	2
NELSTR	3	4	4	6	8	6
NELCHR	5	5	5	5	5	5
NELINT	4	4	4	8	4	1
NELLV	3	3	3	3	4	3
NELLAY	1	1	1	1	5	1
NDI	2	3	3	3	2	3
NSHEAR	1	1	1	3	1	3
JLAW	2	3	4	5	6	7

ITYPE	Element type number.
NELCRD	Number of coordinate data per node.
NELNFR	Number of degrees-of-freedom per node.
NELNOD	Number of nodes per element.
NELSTR	Number of stress and strain components per node.
NELCHR	Number of material property data for the element.
NELINT	Number of 'full' integration points per element.
NELLV	Number of distributed load types per element.
NELLAY	Number of layers of integration through the thickness of the shell element.
NDI	Number of direct stress components.
NSHEAR	Number of shear stress components.
JLAW	Type of the constitutive equation.

this element are defined in pre-integrated form, its use is restricted to linear elastic problems.

In recent PSAM meetings a strong desire has been expressed for the development of advanced element technology needed to address specific SSME applications in an effective manner. Many of these applications involve localized effects which cannot be captured using the classical plate theories. Examples include strong curvature, strong thickness variations or localized mechanical or thermal loading. In principle, continuum theory will always be able to model the proper solution. However, regular continuum elements lack the appropriate deformation modes to model shell-like behavior in a satisfactory way. Recent developments in element formulation suggest that it may be possible to construct continuum elements with enhanced bending behavior that would perform well when degenerated in one direction to form a shell-like element. The development of such an element was proposed by MARC for implementation in the NESSUS code.

2.1.1.6 Solution Strategy and Algorithms

The use of FPI methods in probabilistic finite element analysis involves the repeated computation of the structural response for small perturbations of the random parameters about a given deterministic state. Probabilistic models of realistic structural systems can be quite complex, requiring the analysis of large finite element models parameterized by many random variables. The computational effort expended in the generation of perturbed solutions for these models vastly exceeds that required for all other phases in the analysis. Hence, the ability to efficiently compute the response of the perturbed system is crucial to the viability of the method.

For linear elastostatics, the basic perturbation problem may be expressed as follows. Given the solution to the unperturbed set of finite element equations

$$K u = f \qquad (1)$$

it is desirable to obtain the solution to the perturbed problem

$$\hat{K}\,\hat{u} = \hat{f} \qquad (2)$$

where

$$\hat{K} = K + dK$$

$$\hat{u} = u + du \tag{3}$$

$$\hat{f} = f + df$$

Substitution of these definitions into the equation for the perturbed problem will yield

$$(K + dK)(u + du) = (f + df) \tag{4}$$

$$K\,du = (f + df) - (K + dK)\,u - dK\,du \tag{5}$$

$$K\,du = \hat{f} - \hat{K}\,u - dK\,du \tag{6}$$

Several methods have been proposed for the solution of the problem in this form. A first-order perturbation method may be obtained by neglecting the last term (second-order), and solving for a first-order approximation to du. This approximation can be shown to correspond to the first term in the Taylor series expansion for du. Higher-order perturbation methods are obtained by carrying along additional terms in the Taylor series expansion.

The perturbation strategy adopted in NESSUS/FEM is based on the recovery of the higher-order terms by an iterative process. A suitable algorithm is provided by the recursion form

$$K\,\hat{du}^{(n+1)} = \hat{f} - \hat{K}\,\hat{u}^{(n)} \tag{7}$$

$$\hat{u}^{(n+1)} = \hat{u}^{(n)} + \hat{du}^{(n+1)} \tag{8}$$

This process is equivalent to a modified Newton iteration, and can be shown to satisfy the appropriate consistency condition. The stability of the algorithm will be discussed in Section 2.1.1.7 in some detail.

Many advantages may be derived from adopting this strategy. It should be noted that all computations with the perturbed system can be performed at the element level, and only the resulting element internal force vectors need be assembled into a global vector. This residual force vector must be computed at every iteration step, and provides a direct measure of the quality of the approximation that is used to establish convergence. Furthermore, this approach eliminates the need to compute and store explicit partial derivatives of the element stiffness and load vector, or any assembled form of these quantities. This not only significantly reduces data storage requirements, but also greatly simplifies the coding and validation tasks. The perturbation of geometry and material data is made independent of the element formulation adopted, which allows simple extension of the method to newer element technologies. The overall efficiency of the method can easily be justified on the basis of well-known operation count statistics for large finite element problems.

Additional efficiency improvements are obtainable from recasting the perturbation problem as an iterative process. This allows the implementation of a number of convergence acceleration methods for iterative problems, such as the line search algorithm and quasi-Newton iteration schemes. In particular, significant performance improvements have been demonstrated with the use of either Davidon rank-one secant Newton update or inverse BFGS rank-two update applied to the perturbed elastostatic problem. The present implementation of the line search algorithm does not appear to be very cost-efficient for linear elastostatic problems. This is, in part, due to the fact that it has been implemented as a truly nonlinear line search, since the final goals of the PSAM project call for the extension of the perturbation algorithms to nonlinear problems.

A similar perturbation algorithm for the symmetric eigenproblem with iterative improvement also has been developed and incorporated in NESSUS/FEM. This algorithm differs from earlier eigenproblem perturbation methods by the fact that it has been developed from the start with the intent to tackle realistic structural vibration problems. The problem of properly splitting eigenvalue clusters in the spectrum of the unperturbed problem was identified early on in the algorithm development. A solution was developed, involving a reduced eigenproblem with the dimension of

the multiplicity of eigenvalues in the cluster. Similar formulations typically involve the solution of a larger eigenvalue problem, with dimensions of at least the number of perturbed eigenvalues. An error estimate to account for the effect of a truncated modal representation is also provided. The final form of the algorithm is independent of the method used to obtain the starting eigenpairs, and can be used with any of the modern algorithms for the solution of large symmetric positive-definite eigenproblems.

These perturbation analysis algorithms have been available in NESSUS/FEM since Version 1.0 and have been successfully applied to a broad class of linear problems. It is expected that much of the code developed for the perturbation of linear elastostatics problems will be able to handle weakly nonlinear situations with only minor modifications.

2.1.1.7 Stability Considerations

The iterative perturbation analysis algorithms available in NESSUS have been successfully applied to a broad range of structural problems over the past year. The experience acquired in this testing and validation phase also identified a class of problems for which the iterative process was observed to become unstable with seemingly small values of the perturbation parameter. The problem was first encountered in the analysis of validation problem #2, described in Section 4.0 of Volume III of the PSAM First Year Progress Report. This problem involved the analysis of a thin cantilever beam using bilinear Reissner-Mindlin shell elements (NESSUS element 75) under bending. The stiffness equations for this problem are poorly conditioned, as a result of the enforcement of the transverse shear constraint in the thin limit of the Reissner-Mindlin theory. The problem was observed to be particularly sensitive to geometry perturbations involving changes in element length, which often resulted in loss of stability of the iterative algorithm even for small elongations of the mesh.

A detailed investigation into the nature of the problem was undertaken at MARC, and the major findings are summarized in Appendix B to this report. The investigation concentrated on the analysis of a simpler model problem, involving a mesh of linearly interpolated Timoshenko beam elements. This is the one-dimensional analog of the Reissner-Mindlin plate problem, and exhibits pathological behavior identical to that observed in the plate problem, while offering a much simpler formulation and far more

amenable to a detailed analysis. Early on in the course of the investigation the problem was found to be governed by the characteristics of the assembled stiffness equations. A von Neumann stability analysis of the assembled equations at a typical internal node for the model problem was performed, which provided closed-form expressions for the stability limit in the case of uniform mesh elongation. These stability limits were found to accurately predict loss of stability for numerical experiments involving both beam and shell element discretizations of the model problem. These results can be used to estimate the stability limits for more general beam and plate problems, providing an upper bond for the size of the perturbation parameter that will preserve the stability of the algorithm.

In general, stability will present a concern for the analysis of any problem involving some form of implicit constraint equations in the underlying theory. Stability problems will typically arise whenever geometry perturbations affecting these constraint equations are imposed on the unperturbed problem. Such problems include the analysis of

- o Thin plates and shells allowing shear deformation
- o Incompressible elasticity, e.g., rubber-like materials
- o Strongly anisotropic materials
- o Deviatoric rate-independent plasticity
- o Incompressible Stokes and Navier-Stokes flow

Several of these problems are relevant to SSME applications. It must be noted that alternative formulations based on a Taylor series expansion about the unperturbed system are not immune to the problem. This may be concluded by noting that the speed of convergence of the iterative algorithm is closely related to the error associated with the truncation of the Taylor series. The analysis of the general problem is complicated by the fact that stability is often governed by the lowest deformation modes present in the assembled stiffness equations. Thus, the development of general closed-form results for unstructured, multi-dimensional meshes subjected to non-uniform distortion does not appear to be practical. However, the insight obtained from the analysis of simple model problems can be used to develop "smart" algorithms capable of adaptively adjusting the perturbation size in order to retain good convergence characteristics. These stability considerations further emphasize the need to allow for a

reformulation of the deterministic problem at a point sufficiently close to the design point to obtain a good representation of the limit surface within the stability bounds imposed by the algorithm.

2.1.1.8 Inelastic Algorithm Development

The iterative perturbation analysis algorithms developed for the linear elastostatic problem involve no assumptions on the linearity of the problem, and only minor coding and data management modifications should be necessary to extend these algorithms to situations involving mild nonlinearity. The perturbation database must be extended to include a record of the nodal strain histories, and the finite element code must be modified to carry along in parallel the incremental solution data for all perturbed problems. Perturbations must be allowed on additional types of variables, such as the material's elastoplastic constants. This will involve extensions to both the NESSUS/PRE and NESSUS/FEM modules.

The extension of the perturbation algorithms in NESSUS/FEM to inelastic problems raises important issues, which will affect the development of nonlinear algorithms for the remaining years of PFEM development. Version 1.1 of NESSUS/FEM provides solution algorithms for deterministic linear problems using either a displacement-based or mixed iterative finite element formulation. All development of perturbation analysis algorithms to date has been based on the displacement formulation. Implementation of the displacement method for inelastic analysis will require changes to the internal data storage in the code, in order to retain the element strain history record at the integration points. All data input and reporting of strains and stresses as perceived by the user can still be performed on a nodal basis. An alternate approach suggested by the NASA contract monitor involves the adoption of the mixed finite element formulation for probabilistic analysis, in order to maintain the node-oriented internal data storage currently implemented in the code. This approach would lend itself to a somewhat more elegant implementation of some algorithms, but also involves substantial risk associated with the adoption of less mature finite element technology. Before adopting the mixed approach for probabilistic finite element analysis, the existing perturbation algorithms must be exercised and tested with simple elastostatics problems using the mixed formulation. This step is needed to ensure that no unexpected problems arise from the use of the current solution strategy with the mixed finite element formulation. If the

results of this experiment are positive, then it will be reasonable to consider proceeding with the mixed method for the development of inelastic solution algorithms in NESSUS/FEM.

Further development of inelastic algorithms for probabilistic finite element analysis is awaiting the outcome of the decision on the finite element formulation to be pursued.

2.1.2 NESSUS/FPI Development

2.1.2.1 Eigenvalue Models for Non-normal Distributions

Eigenvalue models for normally distributed, correlated variables (Reference [1]) have been used in the NESSUS code to solve problems involving random fields such as pressure or temperature fields. The NESSUS/PRE module is designed to generate uncorrelated variables based on the covariance matrix of the dependent variables. This is described above in Section 2.1.1.3. The reasons for using the eigenvalue models are:

1. Uncorrelated variables allow fast probability estimation using the NESSUS/FPI module.

2. The number of the significant uncorrelated variables is always less than the number of the correlated variables, and therefore the eigenvalue model is able to reduce the dimension of the random variables entering the perturbation analysis.

To extend the above method to problems involving non-normally distributed, correlated variables, a model has also been formulated [2]. However, the new model is still under investigation and is not yet included in the NESSUS code system. A summary of the method is given in this section. An example involving a highly non-normally distributed variable is given to test the model. The results suggest that, in order to obtain accurate transformed correlation coefficients, higher order terms in the Taylor's series expansion that is used must be retained. Therefore, a more accurate formula relative to the one derived in [2] has been derived and is reviewed in this section.

The eigenvalue model for the non-normally distributed, correlated variables requires two extra steps: the transformation of all variables to the normal distribution space; and, the derivation of the correlation coefficients of the transformed normally distributed variables. The normalization process is defined in (9),

$$F_{X_i}(X_i) = \Phi_i(u_i) \qquad i=1,n \tag{9}$$

where $F_{X_i}(\cdot)$ is the original marginal distribution of the random variable X_i, where $\Phi(\cdot)$ is the normal CDF, and where u_i is a standard normal variate. Note that (9) defines a one-to-one mapping; therefore, X_i may be formulated using the inverse transformation:

$$X_i = F_{X_i}^{-1}(\Phi_i(u_i)) \tag{10}$$

The inverse CDF's, i.e., $F_{X_i}^{-1}(\cdot)$ are available in closed form for such distributions as the Weibull and Type I extreme value distributions. Using (10), the performance function becomes a function of u_i.

We next consider the computation of the correlation coefficients of the transformed variables. Consider two correlated random variables, denoted as X_1 and X_2. The correlation coefficients $\rho_{X_1X_2}$ can be computed as

$$\rho_{X_1X_2} = \frac{E[X_1X_2] - E[X_1]E[X_2]}{\sigma_1\sigma_2} \tag{11}$$

Define the transformation from X_i to u_i as

$$X_i = T_i(u_i) \qquad i=1,2 \tag{12}$$

and define

$$H(u_1,u_2) = X_1X_2 = T_1(u_1)T_2(u_2) \tag{13}$$

Eq. 11 may be expressed as

$$\rho_{X_1X_2}\sigma_1\sigma_2 = E[H] - E[T_1]E[T_2] \tag{14}$$

Using a series expansion method, it can be shown that

$$\rho_{X_1X_2}\sigma_1\sigma_2 = \rho[H_{11} + \tfrac{1}{2}(H_{13}H_{31})] + \tfrac{1}{2}\rho^2H_{22} + \text{H.O.T.} \tag{15}$$

where higher order terms is denoted (HOT) and where

$$H_{ij} = \frac{\partial^{i+j} H}{\partial u_1^i \partial u_2^j} \Bigg|_{u=0} = \frac{d^i T_1}{du_1^i} \cdot \frac{d^j T_2}{du_2^j} \Bigg|_{u=0} \tag{16}$$

and where ρ is the correlation coefficient of the transformed variables u_1 and u_2.

An approximation formula which includes H_{ij} terms up to $i+j=8$ has been derived earlier [2], but was later found to be insufficiently accurate for very non-normally distributed functions, such as the uniform distribution. To improve the accuracy of ρ, a more complete formula with up to 12 terms in the series has been derived and is given in Table 2.2.

In [2], a procedure was formulated to compute H_{ij} using a numerical method. The procedure was demonstrated by using two examples involving lognormal and normal variables. It was found that the series converges rapidly. However, it was not clear how the series would converge if the random variables were strongly non-normally distributed. In the following example a problem involving a uniformly distributed variable is tested to provide information about the rate of convergence of the outlined procedure. This experience is being used to guide the implementation of the procedure into the NESSUS code.

Consider a case where one (say X_1) of the two random variables is normally distributed; then

$$\frac{dX_1}{du_1} = \sigma_1 \tag{17}$$

$$\frac{d^n X_1}{du_1^n} = 0 \text{ for } n > 1 \tag{18}$$

for $u_1 = 0$.

Table 2.2

Series Expansion

$$\rho X_1 X_2 \sigma_1 \sigma_2$$

$$= \rho\,[H_{11} + \tfrac{1}{2}(H_{13}+H_{31}) + \tfrac{1}{8}(H_{15}+2H_{33}+H_{51}) + \tfrac{1}{48}(H_{17}+3H_{35}+3H_{53}+H_{71}) + \tfrac{1}{384}(H_{19}+4H_{37}+6H_{55}+4H_{73}+H_{91}) + \tfrac{1}{3840}(H_{1,11}+5H_{39}+10H_{51}+10H_{75}+5H_{93}+H_{11,1})]$$

$$+ \frac{\rho^2}{2}[H_{22} + \tfrac{1}{2}(H_{24}+H_{42}) + \tfrac{1}{8}(H_{26}+2H_{44}+H_{62}) + \tfrac{1}{48}(H_{28}+3H_{46}+3H_{64}+H_{82}) + \tfrac{1}{384}(H_{2,10}+4H_{48}+6H_{66}+4H_{84}+H_{10,2})]$$

$$+ \frac{\rho^3}{6}[H_{33} + \tfrac{1}{2}(H_{35}+H_{53}) + \tfrac{1}{8}(H_{37}+2H_{55}+H_{73}) + \tfrac{1}{48}(H_{39}+3H_{57}+3H_{75}+H_{93})]$$

$$+ \frac{\rho^4}{24}[H_{44} + \tfrac{1}{2}(H_{46}+H_{64}) + \tfrac{1}{8}(H_{48}+2H_{66}+H_{84})]$$

$$+ \frac{\rho^5}{120}[H_{55} + \tfrac{1}{2}(H_{57}+H_{75})]$$

$$+ \frac{\rho^6}{720}[H_{66}]$$

The approximating series for computing the transformed normal distribution correlation coefficient, ρ, may be derived as (up to the twelfth-order term):

$$\rho_{X_1X_2}\sigma_2 = \frac{\rho}{\sigma_1}\left[H_{11} + \frac{1}{2}H_{13} + \frac{1}{8}H_{15} + \frac{1}{48}H_{17} + \frac{1}{384}H_{19} + \frac{1}{3840}H_{1,11}\right] \tag{19}$$

$$= \rho\left[\frac{dx_2}{du_2} + \frac{1}{2}\frac{d^3x_2}{du_2^3} + ..\right]\Bigg|_{u_2 = 0}$$

where $\rho_{X_1X_2}$ is the original correlation coefficient.

Assume that X_2 is a uniformally distributed variable with a density function defined as

$$f(X_2) = 1 \qquad 0 \le X_2 < 1 \tag{20}$$
$$= 0 \qquad \text{otherwise}$$

Using the normalization scheme given in [1], the relationship between X_2 and the standardized normal distribution function variable u_2 is

$$X_2 = \Phi(u_2) \tag{21}$$

where $\Phi(u_2)$ is the standard normal CDF. Eq. 21 is plotted in Fig. 2.10. Note that a scale factor has been applied to X_2 such that a linear relationship with a slope of one in Fig. 2.10 represents a standard normal distribution. Therefore, the uniform distribution, according to Fig. 2.10 behaves in a significantly non-normally distributed fashion for $|u_2| > 1$.

Using the numerical algorithm from [1], thirteen sets of data are obtained as follows in Table 2.3:

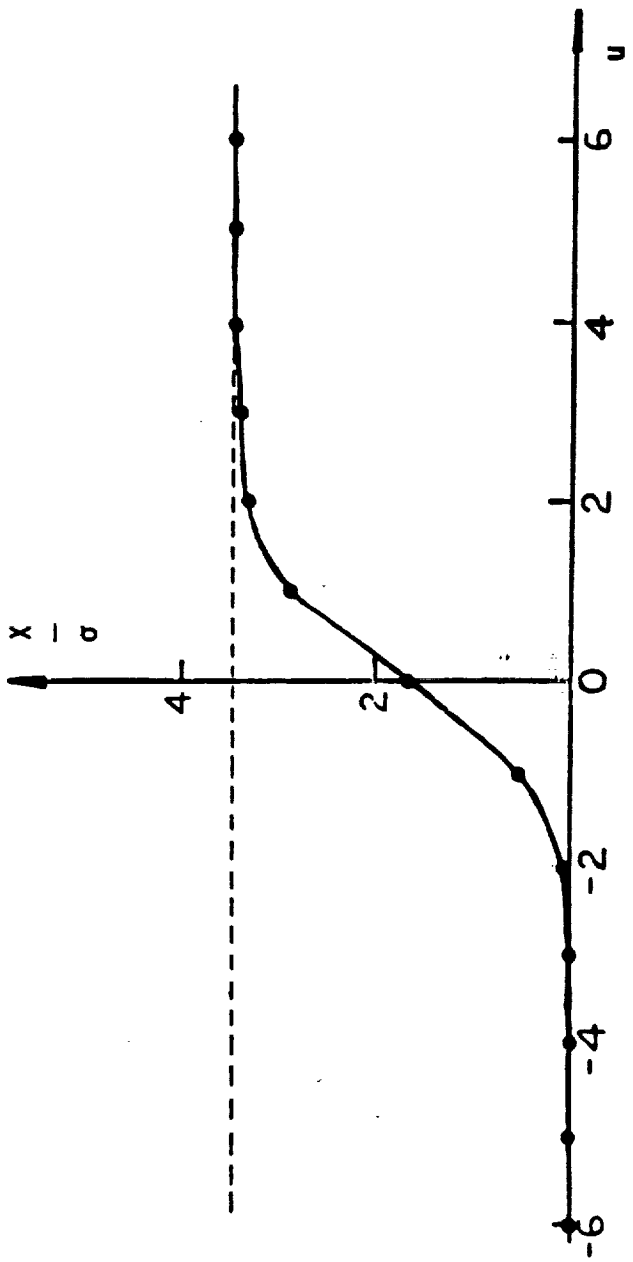


Fig. 2.10 Transformation from a Uniform Distribution to a Standardized Normal Distribution

Table 2.3

Results of Equivalent Normalization of Uniform Distribution

Set	u_2	X_2
1	-6	.990E-09
2	-5	.287E-06
3	-4	.316E-04
4	-3	.00135
5	-2	.00227
6	-1	.158
7	-0	.5
8	1	.841
9	2	.977
10	3	.9986
11	4	.999968
12	5	.999999713
13	6	.999999999

The next step is to construct a twelfth-order polynomial denoted as

$$X_2 = \sum_{n=0}^{12} A_n u_2^n \tag{22}$$

The required derivatives for computing ρ are

$$\left.\frac{d^n X_2}{du_2^n}\right|_{u_2=0} = A_n \cdot n! \tag{23}$$

where n = 1, 3, 5, 7, 9, and 11.

By solving thirteen simultaneous linear equations, the coefficients A_n can be found. Using Eq. 23 and Eq. 19 the approximation solution is

$$\rho_{X_1X_2} = 1.36225\rho[1. - 0.4380 + 0.2235 - 0.0871 + 0.0214 - 0.0024] \tag{24}$$

The final results are as follows

$$\begin{aligned}
\rho \text{ (Sixth-order)} &= 0.9345\ \rho_{X_1X_2} \\
\rho \text{ (Eighth-order)} &= 1.051\ \rho_{X_1X_2} \\
\rho \text{ (Tenth-order)} &= 1.020\ \rho_{X_1X_2} \\
\rho \text{ (Twelfth-order)} &= 1.023\ \rho_{X_1X_2}
\end{aligned} \tag{25}$$

The exact solution for this particular case is available [3] and is

$$\rho \text{ (Exact)} = 1.023\ \rho_{X_1X_2} \tag{26}$$

Eqs. 25 and 26 show that the Taylor's series from (24) converges quite slowly. It needs to be pointed out that this example is considered to be an extreme case to test the robustness of the algorithm.

2.1.2.2 FPI Validation Studies

The original FPI (Fast Probability Integration) code using an algorithm developed by Wu [4] was modified to become the NESSUS/FPI code. In [4], the performance of the algorithm is assessed by six examples; some examples are considered the worst possible cases. The results indicate that the algorithm is able to provide accurate or reasonably good point probability estimates. In all cases, the results are significantly better than a widely-used FPI method: the first-order reliability analysis [5].

Chang [6] has investigated the performance (accuracy and efficiency) of the FPI algorithm for computing structural reliability. Thirteen examples have been used to test the FPI accuracy. Many of the examples had nonlinear performance functions with non-normal random variables. The maximum number of random variables in the examples is twenty. The results indicate that the FPI algorithm provide good probability estimates. The errors in the point probability estimates are less than or near 5% in twelve examples which are typical of mechanical design problems.

The only example which results in large error is the same problem investigated in Reference [4]. The performance function of the example is a linear combination of the identical and independent random variables. Each variable is chi-square distributed with one degree of freedom. The distribution is apparently highly non-normal. Its density function has a shape similar to an exponential density function (i.e., Aexp(-Ax) where A is a positive constant) which has only one tail. It seems obvious that this distribution can not be fitted well by a symmetric bell-shaped normal curve. This test example indicates that the accuracy of the current NESSUS/FPI code is limited by the normality of the random variables. However, in probabilistic structural analysis, non-normal engineering variables are commonly modeled using the standard distributions such as the lognormal and the extreme value distributions. Using the FPI algorithm, these distributions can usually be fitted very well (in the least-squares sense) by the three-parameter normal distributions.

The FPI code has also been compared with a code based on the second order reliability methods [7]. Three examples taken from Chang's report were tested. The comparison of computed probability estimates suggest that there are no significant differences in accuracy. The computational efficiencies were also compared by assuming that the computational efficiency for the first-order reliability analysis should be approximately equal using the two codes.

The comparison of the computer time seems to confirm that, at least for linear performance functions, NESSUS/FPI is faster than the second order reliability methods, especially for a large number of variables (in the test examples, the maximum number of random variables, N, is 20). The reason is believed to be that the second order methods needs to compute all the second order derivatives of the performance function in the transformed "standardized normal (u) space", whereas the NESSUS/FPI algorithm considers only part of the second order derivatives of the performance function in the X space. The advantage of using the NESSUS/FPI algorithm is significant, since the computational effort required by the NESSUS/FPI is of order N, while that required by the second order methods are of order N^2. Moreover, since it is very inefficient to establish a "complete" quadratic response function in a typical NESSUS analysis, it seems more likely that the established response function will be either linear or

incomplete quadratic. In such cases, the NESSUS/FPI algorithm is particularly efficient because it does not require the computation of the second order derivative.

The NESSUS/FPI algorithm has also been used to demonstrate how to compute the probability of instability of a dynamic system [8]. The system is represented by an n-th order linear differential equation. By assuming a solution of the form exp(st), a characteristic polynomial equation is obtained where the coefficient are random functions of the random variables. A root s with a positive real part means that exp(st) becomes unbounded and the system is unstable. A procedure based on the FPI algorithm is developed and demonstrated using an example involves a six degree polynomial with two random variables. For comparison purposes, a Monte Carlo solution is obtained. The result shows that FPI is accurate and is far more efficient than the simulation method.

Other NESSUS/FPI validation exercises include the solution of the NESSUS validation test cases 1 and 2 in which good agreement between FPI and Monte Carlo are obtained.

A general conclusion drawn from the results of the numerous examples is that the NESSUS/FPI is consistently able to provide accurate results so long as the expansion point is the most probable point. When the most probable point can be located (by iteration), good results can usually be expected even with linear approximation of the performance function.

2.1.2.3 FPI Accuracy/Improvement Studies

A number of studies on the FPI algorithm were conducted at the University of Arizona. The studies focused on approximation functions within NESSUS/FPI which have been suspected of producing errors in the resulting probability estimates. Modifications to NESSUS/FPI have been made to improve the performance of the code and include:

1. A new gamma function has been introduced. This function representation has about eight significant figures for accuracy and is a significant improvement over the polynomial approximation previously used.

2. Changes were made in calculations of the extreme value distribution (EVD) parameters to provide ten-place accuracy.

3. The polynomial approximation to the inverse normal CDF has been replaced by the secant method with a significant improvement in accuracy.

4. All distribution parameters are now computed at the beginning of the program instead of within the subroutines.

5. The secant method is used to compute the Weibull shape parameter. Using this method and the new gamma function should improve accuracy of both Weibull parameters.

Details of these changes are reported in Appendix C.

Comparison of the old and new code for NESSUS/FPI showed small changes in the results, generally less than 5% for all thirteen test examples. However, the new code accuracy has been achieved with no significant loss of efficiency and is, therefore, being incorporated in the next release of NESSUS/FPI.

In addition to the above numerical improvements, there are three new distributions which have been added to NESSUS/FPI. The new distributions are:

1. The Frechet distribution Type 2 asymptotic distribution of extreme values from an initial lognormal distribution and, in general, an initial distribution having a polynomial tail in the direction of the extreme.

2. Truncated Weibull distribution.

3. Truncated normal distribution.

The truncated distributions are included for modeling distributions of material axes for the turbine blade verification problem. Other distributions already in the code are the normal, lognormal, Weibull, Type 1 extreme value, maximum entropy, chi-square, and NESSUS. The NESSUS distribution is a polynomial of a normally distributed variable.

2.1.2.4 Confidence Band Estimation

The basic goal of confidence band estimation, in the context of the NESSUS analysis, is to quantify the confidence on the accuracies of the probability estimates for the response functions. The basic assumption for the methods is that the response functions are derived from the NESSUS perturbation data base. The approach is to treat the distribution parameters of the input random variables as random variables, and then create the CDF of the response function CDF. This strategy is the essence

of the Beysian approach to parameter estimation. Four approximation methods are identified for estimating the confidence (or error) band of the cumulative distribution function of the response function. The most suitable one is identified and included in the new version of the NESSUS/FPI code.

Let Z denote the response variable, and Z(X) denote the response function where X is a vector of the input independent variables. In the NESSUS/FPI, Z(X) is a polynomial function:

$$Z = Z(X) = a_0 + \sum_{i=1}^{N} a_i X_i + \sum_{i=1}^{N} b_i X_i^2 \qquad (27)$$

where N is the number of independent random variables. In general, several polynomial equations may be required to ensure sufficient accuracy of the function over a wide range of Z. Ideally, one polynomial should be established for a selected Z value.

The basic assumption in the response function for the confidence interval estimation is that for a given Z, the best estimate Z(X) (derived using the best estimates statistics of X) remains valid within the confidence band. In general, Z(X) is different for different distribution parameters set because the most probable point, which is used to define Z(X), is a function of the distributions. However, the assumption is valid when Z(X) is actually a first or second degree polynomial. For highly nonlinear Z(X) function, the assumption is a reasonable one so long as the variabilities of the significant random variables are not very large.

There are two basic types of uncertainties in a NESSUS/FPI-generated response function: (1) physical uncertainty and (2) model uncertainty. Physical uncertainty is the uncertainty associated with physical phenomena which are inherently random. In the NESSUS analysis, this uncertainty is accounted for by treating the input variables as random variables or random fields. Model uncertainty includes parameter uncertainty, uncertainty in the statistical distribution model, response function model error, etc. The approach adopted in this study concentrates on the variabilities of the input variables.

Assume that X is a normally distributed random variable with mean μ and standard σ deviation. Given a sample, n, the sample mean, $\bar{X}$, is a normal variable with mean and standard deviation of

requires major code modification and developement effort. Method 3 is accurate for large samples, but the full simulation is extremely inefficient. Method 4 is accurate and is consistent with the current NESSUS/FPI approach, i.e., the CDF of Z(X) is computed by using the FPI method for a given set of statistical parameters. In terms of the computational efficiency, Method 4 is inefficient relative to Methods 1 and 2 but is much faster than Method 3. Overall, Method 4 was considered accurate with satisfactory efficiency, therefore, it was selected and has been incorporated in the NESSUS/FPI code.

An example has been taken from that proposed in Appendix D as a means to test the confidence band estimation algorithm in the NESSUS/FPI code. The response function Z is a function of two normal variables X and Y,

$$Z = X - Y$$

The statistics are,

For X	For Y
$n = 20$	$n = 20$
$\overline{X} = 10$	$\overline{Y} = 5$
$s_X = 2$	$s_Y = 1$

The sample means and the sample standard deviations are defined as the best estimates. Using Eqns. 28 and 29, the COVs for the means of both X and Y are 0.0447; the COVs for the standard deviations of both X and Y are 0.162. By entering these parameters into the NESSUS/FPI, the 90% and 95% confidence bands of the CDF of Z were generated. The result is shown in Fig. 2.11. The Monte Carlo sample size is 5,000.

2.1.2.5 Monte Carlo Methods

Monte Carlo simulation has been usually considered to be a last resort for solving a major simulation problem because of its high cost for accurate results, especially in the tails of the distributions. However, recent developments of new and efficient algorithms have made Monte Carlo more attractive.

A study of several Monte Carlo simulation algorithms has been conducted at the University of Arizona for the PSAM project. Two

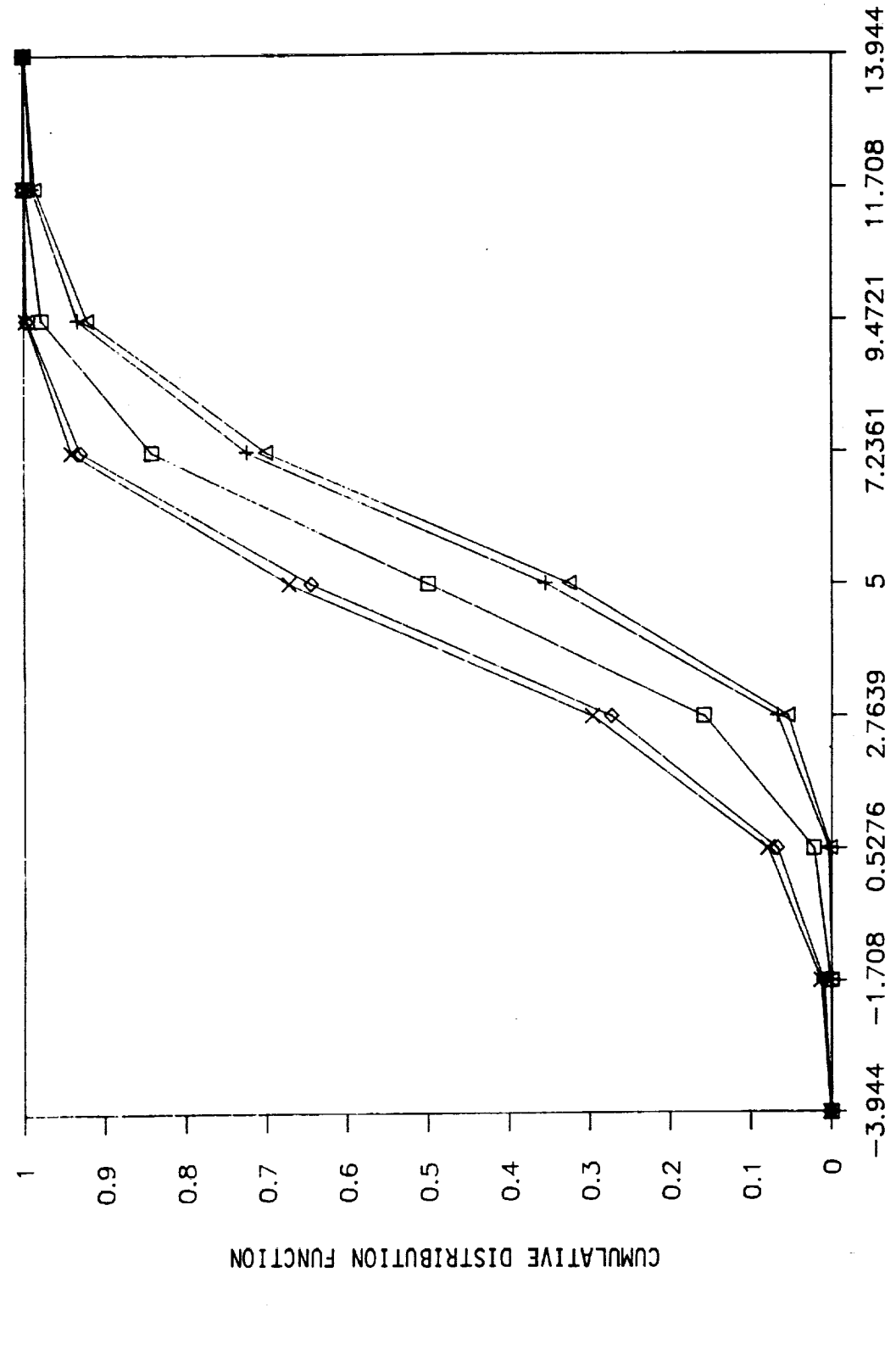


Fig. 2.11 Confidence Interval Estimation

$$\mu_{\bar{X}} = \mu \qquad s = \sigma/\sqrt{n} \tag{28}$$

The variable $(n-1)\ s^2/\sigma^2$ has a chi-square distribution with n-1 degree of freedom. If this chi-square distribution is approximated by a lognormal, then the distribution of s will be a lognormal. The statistic of s can be approximated as

$$\mu_s = s \qquad \sigma_s = \sigma/\sqrt{[2(n-1)]} \tag{29}$$

For the NESSUS/FPI confidence band estimation, we assume that each $X_i (i=1,N)$ is characterized by its mean and standard deviation. We further assume that the statistical distribution is normal for the mean, and is lognormal for the sample standard deviation. These assumptions about the statistical distributions of the parameters are exact only when X is normal. The actual distributions usually do not follow available standard distributions and the CDF's cannot be defined in closed forms.

The required input data for the confidence band estimation are the statistics (the means and the COVs (coefficient of variation = standard deviation/mean)) for the means and the standard deviations of all X_i's. Note that the input statistics may be estimated by using Eqs. 28 and 29 where the actual statistics may be replaced by sample statistics. However, the statistics may also be estimated using other statistical methods or engineering judgement. This input format is more flexible since it does not require that the sample sizes be defined. However, the input statistics must be prepared before the estimation process.

Four methods are considered:

1. First Order Error Bounds

Assume that Z(X) is linear and each X_i is normally distributed. For each Z(X), there is a best-estimate most probable point of X. The best estimate CDF of X_i, denoted as F_i, is determined using the most probable point for each X_i. At F_i, X_i can be written in terms of the mean and standard deviation by inverting the CDF,

$$X_i = \sigma_i \Phi^{-1}[F_i] + \mu_i \tag{30}$$

where σ_i and μ_i are random variables. Upon substitution of each X_i into Eq. 30, Z(X) can be expressed in terms of $Z(\underset{\sim}{\mu},\underset{\sim}{\sigma})$.

By further assuming that Z(X) has a normal or a lognormal distribution, closed form solutions for the confidence bounds were derived by Wirsching and are included in Appendix D.

2. FPI algorithm

Assume that Z(X) is linear. The CDF of Z(X) = z, in terms of the standard normal variate, u, can be formulated as

$$u = \frac{a_o + \Sigma a_i \mu_i}{\sqrt{(\Sigma a_i^2 \sigma_i^2)}} \tag{31}$$

where μ_i and σ_i are the equivalent normal parameters of X_i based on the Rackwitz-Fiessler algorithm [9]. Eq. 5 is a safety index formulation based on the first order reliability method.

Note that the equivalent normal parameters are functions of the CDF, F(X), the PDF (probability density function), f(X), and the most probable point X_i. Let

$$\theta_i = (\mu_i, \sigma_i) \tag{32}$$

Because F(x) and f(x) are functions of Φ, therefore, u can be expressed as

$$u = \text{function } (\underset{\sim}{\theta}) \tag{33}$$

The limit state or performance function can be formulated as

$$g(\underset{\sim}{\theta}) = u - u_o \tag{34}$$

The following is a proposed FPI iteration algorithm for estimating the CDF of u for a selected response function value Z = z:

1. Select a u_o.
2. Guess the design point of the distribution parameters, $\underset{\sim}{\theta}$.
3. Compute the equivalent normal parameters of the random variables, $\underset{\sim}{\theta}$.

4. Define the distributions of X using the most probable point of θ.

5. Guess the most probable point of the basic variable X.

6. Compute equivalent normal parameters for non-normal X.

7. Compute the most probable point of X and the CDF of $Z(X) = z$

8. Go to step 3; repeat until the most probable point of X or the CDF of Z(X) is stabilized.

9. Compute the most probable point of θ and the CDF of $g(\underset{\sim}{\theta}) = 0$

10. Go to step 2; repeat until the most probable point of $\underset{\sim}{\theta}$ or the CDF of $g(\underset{\sim}{\theta}) = 0$ is stabilized.

Note that the above procedure requires nested iteration loops. Step 3 to step 8 constitute the inner FPI loop for a selected θ set. Steps 2, 9 and 10 constitute the outer loop for finding the most likely θ set.

3. "Full" Monte Carlo Simulation

This method is conceptually more straight-forward. It requires the following steps:

1. Generate samples of $\underset{\sim}{\theta}$ sets, $\underset{\sim}{\theta}_j$, $j = 1, J$

2. For each θ, generate a set of X_k, $k = 1, K$

3. Compute, using X_j, the response function value, Z_k, $k = 1,K$

4. For each $\underset{\sim}{\theta}_j$, compute the CDF of $Z(X)=z$, denoted as $(CDF)_j$, $j = 1, J$

5. Using samples of (CDF), construct CDF of u.

This last procedure is expected to be extremely time-consuming because it requires the generation of "J times K" samples of Z(X) values.

4. Mixed Approach - Combination of Monte Carlo and FPI

This approach combines the above methods (2) and (3). The difference between this approach and the previous approach (Method 3) is that after a set of X_j is generated, the FPI routine is used to compute each $(CDF)_i$.

The methods can now be compared. Method 1 captures the essence of statistical uncertainty and is the most efficient. However, the accuracy of Method 1 is limited by the distributional assumptions. Further improvement is needed for this fast algorithm. Method 2 has the potential to be both fast and accurate, however, it is the most complicated and

computer programs based on the conventional Monte Carlo algorithm and one based on variance reduction using antithetic variables were written at the University of Arizona. Other efficient Monte Carlo schemes are still being evaluated. The work to date is summarized in Appendix E.

2.1.2.6 Integration with NESSUS/FEM

This section summarizes the study of an FPI iteration procedure which was intended to be used to integrate the FPI and the FEM modules. The procedure was used successfully to solve several selected problems. At the end period of this study, however, a new and potentially more efficient method was formulated which seems to be most suitable for constructing the CDF of a response function. The newly-developed method and the iteration procedure are summarized in the next section (2.1.2.7). The procedure described in the present section is useful for computing a point CDF. For creating the entire CDF, the present procedure may ultimately be replaced by the new procedure. However, the new procedure is based on the present study, and many key concepts discussed in this section remain valid for the new procedure.

The integration of the NESSUS/FPI and the NESSUS/FEM is based on the concept of successive linear/quadratic approximation algorithm which was identified in the first year of this project [5]. The goal is to expand or perturb the performance function about the most probable point. Note that in the field of structural reliability analysis, where the goal is to find the probability of failure estimate, the most probable point is called the "design point". The algorithm which is summarized in the following has been used to test several examples with success.

The iterative algorithm has been established as follows:

1. Identify critical dependent variable (stress,frequency,...)

2. Select values for dependent variable. (e.g., mean, mean + 10% of mean)

3. Using the NESSUS/FEM module, compute the perturbation solutions about an initial guessing most probable point. Initially this can be chosen as the mean values. However, a good initial guess of the most probable point will accelerate the iteration procedure.

4. Establish linear/quadratic response surface from the perturbation solutions using the least-squares method.

5. Compute the most probable point, using the NESSUS/FPI, for the selected value of dependent variable.

6. Compute the CDF and evaluate accuracy (based on the successive CDF values or the most probable point values).

7. Use the new most probable point and go to step 3, until the solution converges.

Experience with this algorithm has indicated that the solution can usually be found in about three iterations.

An example is now presented to illustrate the above iteration procedure for integrating FEM with FPI. The example is a simplified version of the NESSUS validation test case 2 from the first year annual report. The finite element model is illustrated in Fig. 2.12. There were initially ten random variables: five correlated loadings, width, length, thickness, base spring and modulus of elasticity. By assuming that the width is deterministic, the "exact" root stress becomes:

$$S = LP/t^2 \tag{35}$$

in which P is a load random variable; L is the length and t is the thickness. The mean value of S is approximately 3500 psi.

In order to illustrate more clearly the iteration procedure, it is assumed further that t is a deterministic variable and L an P are normally distributed. Note that none of the above assumptions is required for the NESSUS solution.

Define the "reduced variables" of L and P as

$$u_1 = (L - L_{mean})/L_{std.\ dev.}$$
$$u_2 = (P - P_{mean})/P_{std.\ dev.} \tag{36}$$

Using Eq. 36, L and P can be expressed as functions of u_1 and u_2, respectively. Substituting L and P into (35), one can plot the contours of constant stress (iso-stress)in a two dimensional u space as shown in Fig. 2.13. The reason for using the u coordinate system is that the joint

Plate Finite Elements

9 Random Variables

- Loads, P1 to P5
- Length, L
- Thickness, t
- Base Spring, K
- Modulus of Elasticity, E

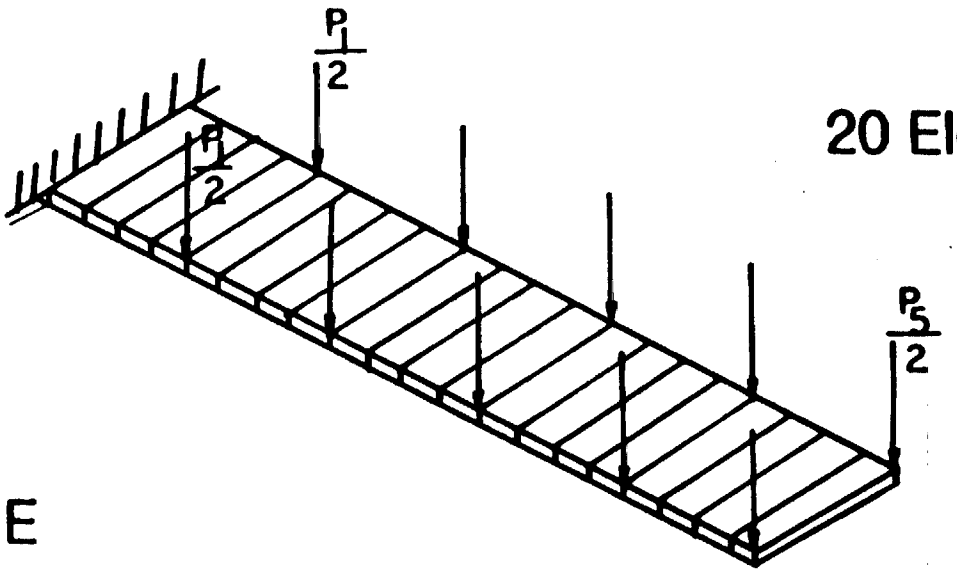


Fig. 2.12 The Test Example

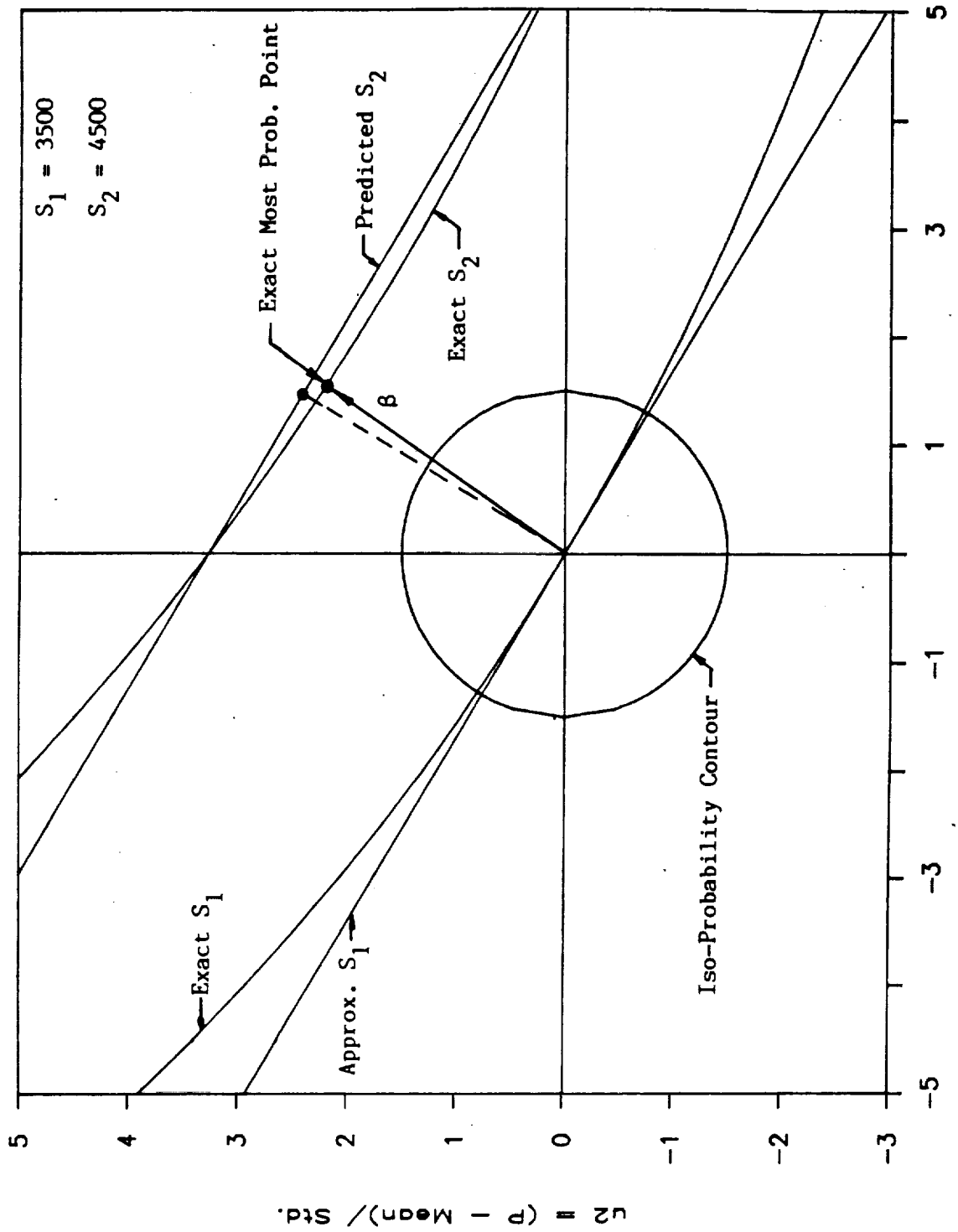


Fig. 2.13 First Iteration Used to Obtain the Most Probable Point

probability density function is rotationally symmetric. The most probable point is, therefore, easily identified as the point on a iso-stress curve which is nearest to the origin.

The problem is defined as follows: Find the most probable point (and the CDF) for S_2 = 4500 psi by starting at S_1 = 3500 psi. Fig. 2.13 shows the result of the first iteration. Initially the linear approximation of S is based on the mean values of P and L which corresponds to the origin. A "predicted" iso-stress curve (S_2 = 4500 psi) can be defined using the mean-derived linear equation. The predicted S_2 curve, which is parallel to the approximated S_1 curve, deviates from the exact S_2 curve because S is actually a nonlinear function of P and L. However, this first iteration leads one to a region close to the exact most probable point. Using the predicted most probable point as a new expansion point, a second iteration results in an accurate prediction of the most probable point as shown in Fig. 2.14. No more iteration is required.

For S^2 > 4500 psi, the volume under the joint probability density function surface is concentrated near the most probable point. The first order reliability analysis gives the following result:

$$P(S > 4500 \text{ psi}) = \Phi(-\beta) \tag{37}$$

where β is the minimum distance defined by the most probable point.

The above procedure can be applied to several values of S in order to eatablish the entire CDF. In the following, the procedure will be applied to integrate the FPI and the FEM. The test problem is the NESSUS validation test case two of which the width is assumed to be deterministic. The results of the first iterations at three stress levels (2600, 3500 and 5400 psi) are shown in Fig. 2.15.

The purpose of Fig. 2.15 is to show the algorithm for integrating the NESSUS/FEM and the NESSUS/FPI. The finite element model consists of twenty plate elements (NESSUS element 75). The difference between the analytical and the NESSUS solutions is about 3%. In order to show the effect of successive linear approximation, a "calibrated exact" CDF is used to match the mean solutions.

The first perturbation was taken about the mean values of the independent variables. Two more FEM perturbations were taken about S =

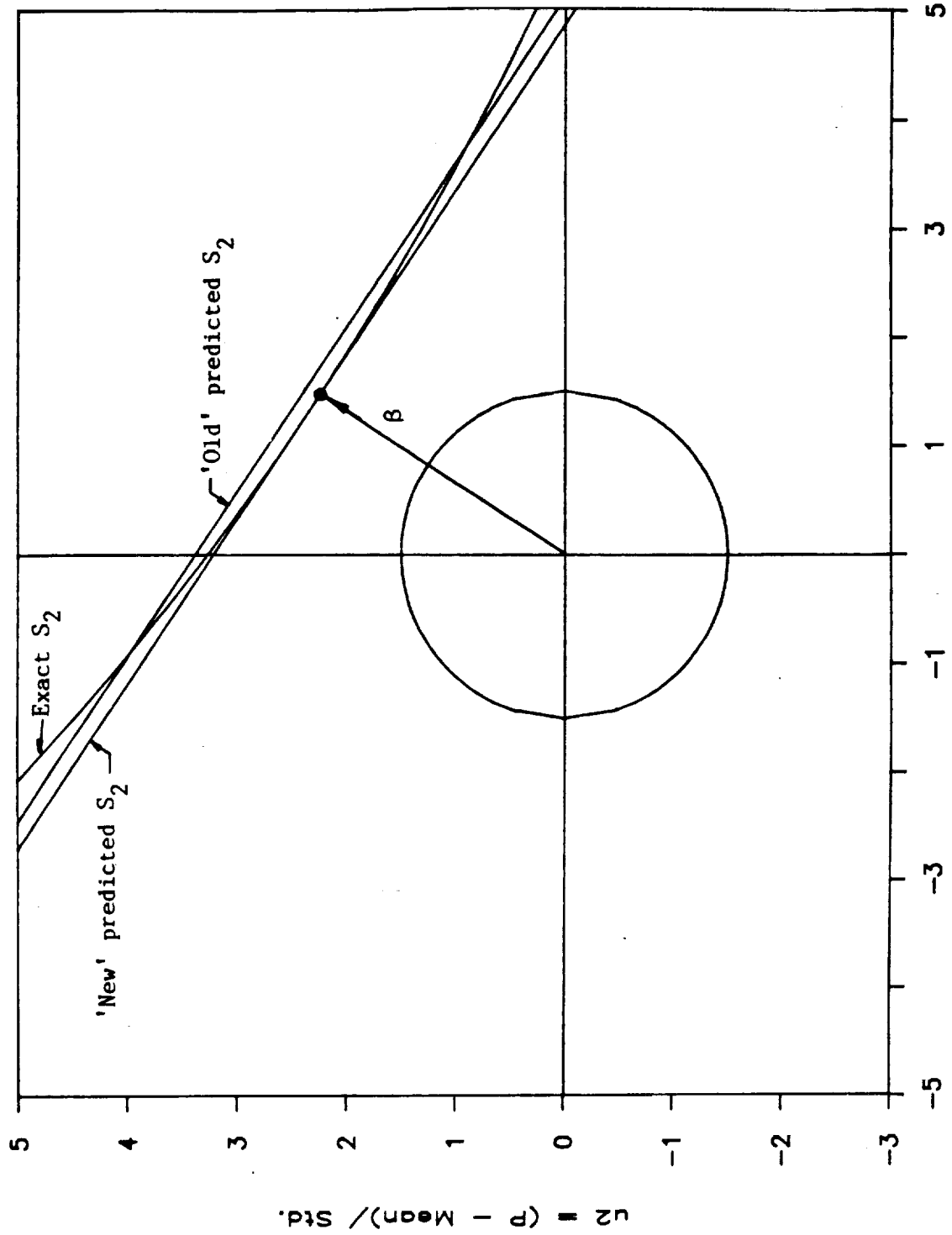


Fig. 2.14 Second Iteration Used to Obtain the Most Probable Point

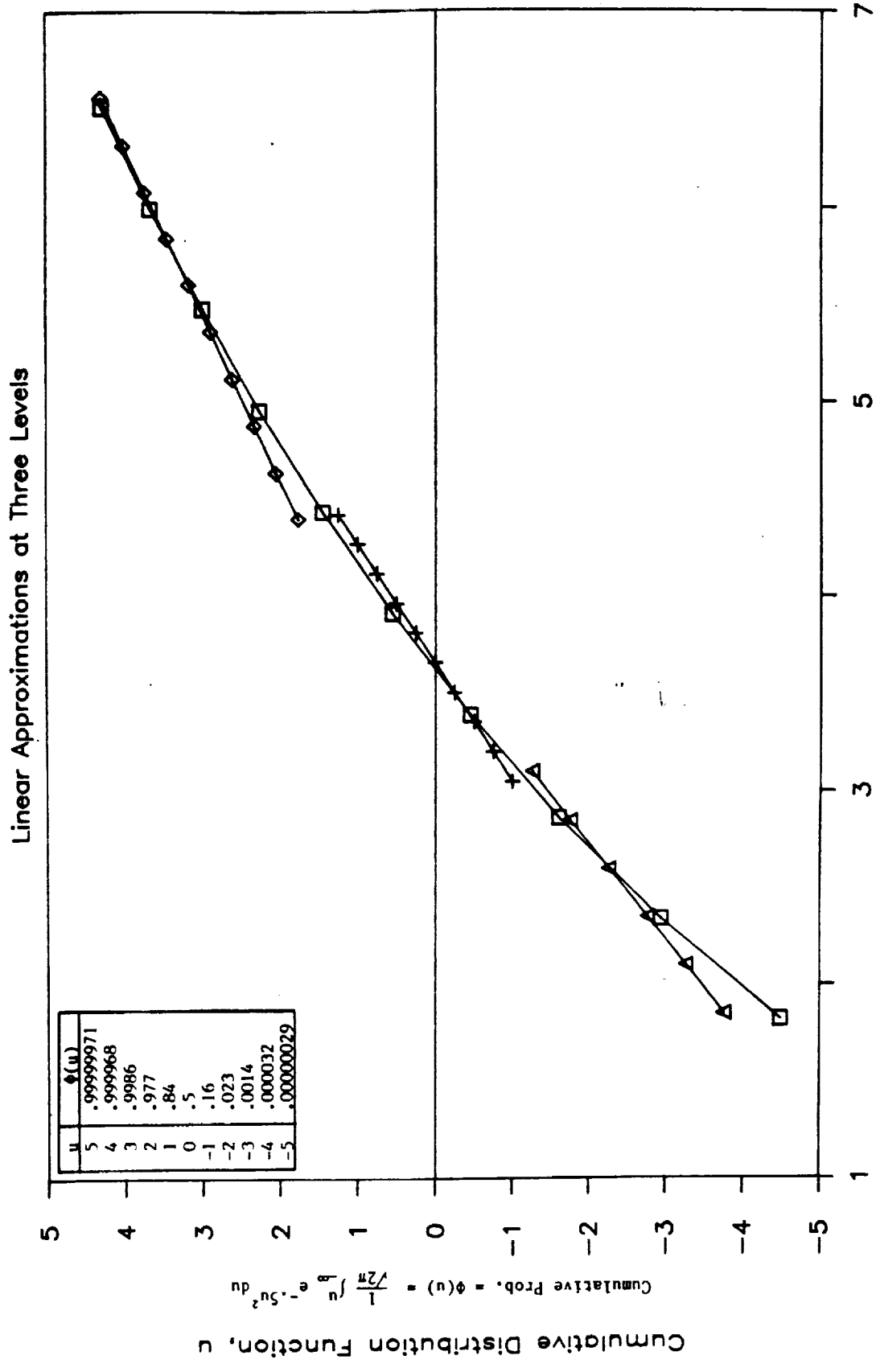


Fig. 2.15 Example - NESSUS/FEM and NESSUS/FPI Integration (Stress)

2600 and S = 3500 using the predicted most probable points. It is shown that the CDF values are accurate "locally" around the approximation points.

Figure 2.16 shows the result of an analysis for the tip displacement of the same cantilever beam. The goal was to compute the CDF at 1.2 inches. The result of the first iteration indicates a significant improvement at the region around 1.2 inches.

It should be noted the above results were obtained using "small" perturbations (0.05 or 0.1 standard deviation for the independent random variables). The reason was to estimate the first order sensitivities more efficiently. It is noted also that the update of the most probable points in the NESSUS/FEM input data deck were done manually. The updated "correlated" nodal loads were being computed using the most probable point values of the "uncorrelated" loads (which means that the eigenvectors generated using the NESSUS/PRE module must be used to update the NESSUS/FEM data). This computational procedure needs to be considered carefully in designing the user-friendly expert system - the NESSUS/EXPERT module.

2.1.2.7 A New CDF Estimation Method

This section summarizes a new CDF estimation method. This method, if proved to be more effective for estimating the CDF of the response function, will replace the one described in the previous section (2.1.2.6). However, since the new method was developed in the last period of the second year PSAM efforts, further detailed study of the method is required to validate the method. A preliminary discussion on the method is given in Appendix F where the formulation of the method and a simple example are included. By using a procedure which corrects the error of the response function at the most probable point, it is shown that the new procedure has the potential to significantly improve the NESSUS solution efficiency by reducing the requirements on the perturbation solutions.

The procedure based on the new method for integrating the NESSUS/FEM and the NESSUS/FPI modules is as follows:

1. Construct first (can be extended to second) order approximation of the response function Z(X) about the mean values. NESSUS/FEM module is used to generate response function sensitivities or perturbation solutions.

2. The reponse function is established using the least-squares routine in the NESSUS/FPI.

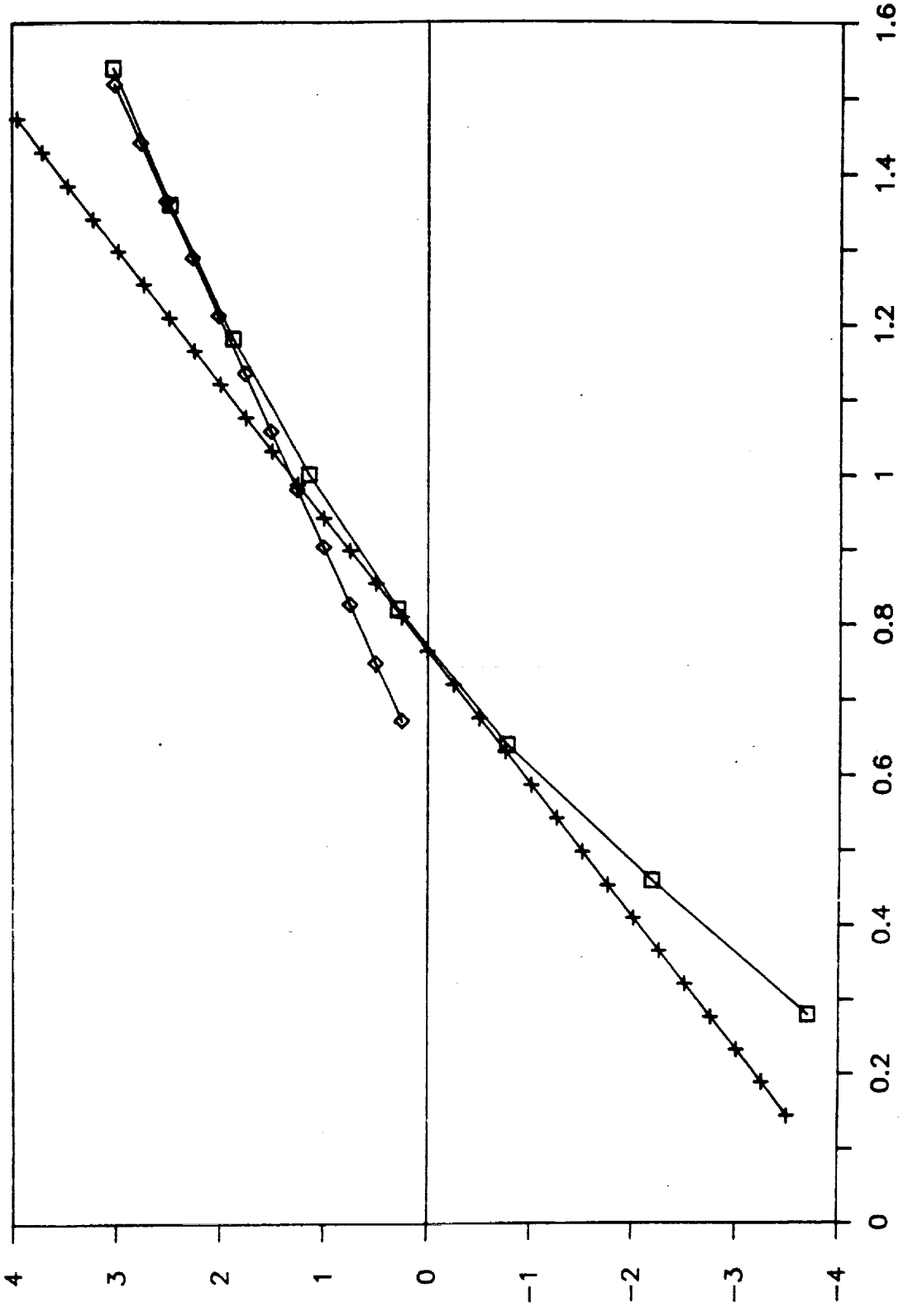


Fig. 2.16 Example - NESSUS/FEM and NESSUS/FPI Integration (Displacement)

3. Using the response function of step 2, a CDF can be constructed. (This CDF is, in general, not sufficiently accurate at the tail regions of the distribution.)

4. Select a CDF value from the result of step 3; find the corresponding "predicted" response value, Z_p.

5. At $Z = Z_p$, compute the most probable values of X, X_p, using the NESSUS/FPI module.

6. Using the NESSUS/FEM, compute the "exact" response function value, Z_e, at the most probable point, X_p. Z_e is the "corrected" value for the selected CDF value defined in step 4.

7. Compare Z_e and Z_p. If the difference is small (say, less than 20 %) go to step 3 and select another probability level. If the difference is large, go to step 1 and replace the mean values of X by the X_p values.

The significant difference between the present procedure and the one presented in the previous section is that the present procedure fixes a CDF value and looks for the accurate corresponding response function value, whereas in the previous procedure, a response function value is fixed and the CDF value is found using an iteration procedure. Thus, the present procedure relies more on the additional deterministic solutions while the previous procedure relies heavily on the additional sensitivity analyses. Since the sensitivity analyses require more computational efforts than the deterministic analyses, it seems resonable to expect that the new procedure will be more efficient.

2.1.2.8 NESSUS/FPI Code Validation Studies

A test plan for validating the first year probabilistic finite element code was included in the First Year Annual Report (Vol. III, Section 4). It consisted of nine validation problems which were designed to test a variety of capabilities of the NESSUS code. The exact solutions, in terms of the probability distributions or the probability of exceedance, have been obtained for the first five validation problems. The results which are summarized in the following are presented in a format compatible with the NESSUS/FPI output. "Exact" solutions are obtained using the Monte Carlo simulation if no closed form solution is available. These exact solutions are to be compared with the NESSUS solution to validate the code as well as the solution algorithm (i.e., FPI iteration algorithm).

The exact solution for the validation problem 1 was included in the First Year Annual Report. The problem addressed is a cantilever beam subjected to static loadings, $P_i(i=1,5)$ (see Fig. 2.17). The loadings are correlated random variables. Other random variables include Young's modulus, length, thickness, width, base spring and yield strength. The expected output of the NESSUS solution include the CDFs of the maximum stress and the tip displacement, and the probability that the stress will exceed the yield strength. The type of finite element used in NESSUS is beam element 98.

Problem 2 is similar to problem 1 except that the plate element is used and the thickness of the beam is reduced. Because of the reduced thickness, the nodal loads were changed from 20 lbs to 0.1 lbs. Figs. 2.18 and 2.19 summarize the results of the CDF of the maximum stress, the CDF of the tip displacement and the probability that the stress exceeds the yield strength.

The goal of the validation problem 3 is to validate the NESSUS eigenvalue solution algorithms. The cantilever beam defined in problem 1 is used again. The response functions tested are the first three bending frequencies in each lateral direction. The analytical solutions for the frequencies in the X and Z directions modes were used to derive the exact CDFs. Figure 2.20 and Table 2.4 summarize the results for the X direction; Fig. 2.21 and Table 2.5 summarize the result for the Z direction.

Validation problems 4 and 5 addressed a rotating beam as illustrated in Fig. 2.22. The random variables are: mass density, length, Young's modulus, thickness and width. Problem 4 tests the beam element, and problem 5 tests the plate element. The response function tested are the tip axial displacement and the first bending frequency in the Z direction. The analytical solutions are the same for both problems. In the original test plan, the beam was fixed at the rotation center. To represent a turbine blade configuration more closely, the inner radius (measured from the center of rotation to the "fixed" end of the beam) was defined to be 4.237 inches. Analytical solutions were revised and used to generate exact solutions using Monte Carlo simulation. Figures 2.23 and 2.24 summarize the results for the tip displacement and the fundamental bending frequency.

The NESSUS code validation is still in progress, and MARC will run the NESSUS/FPI code and compare results with these "exact"

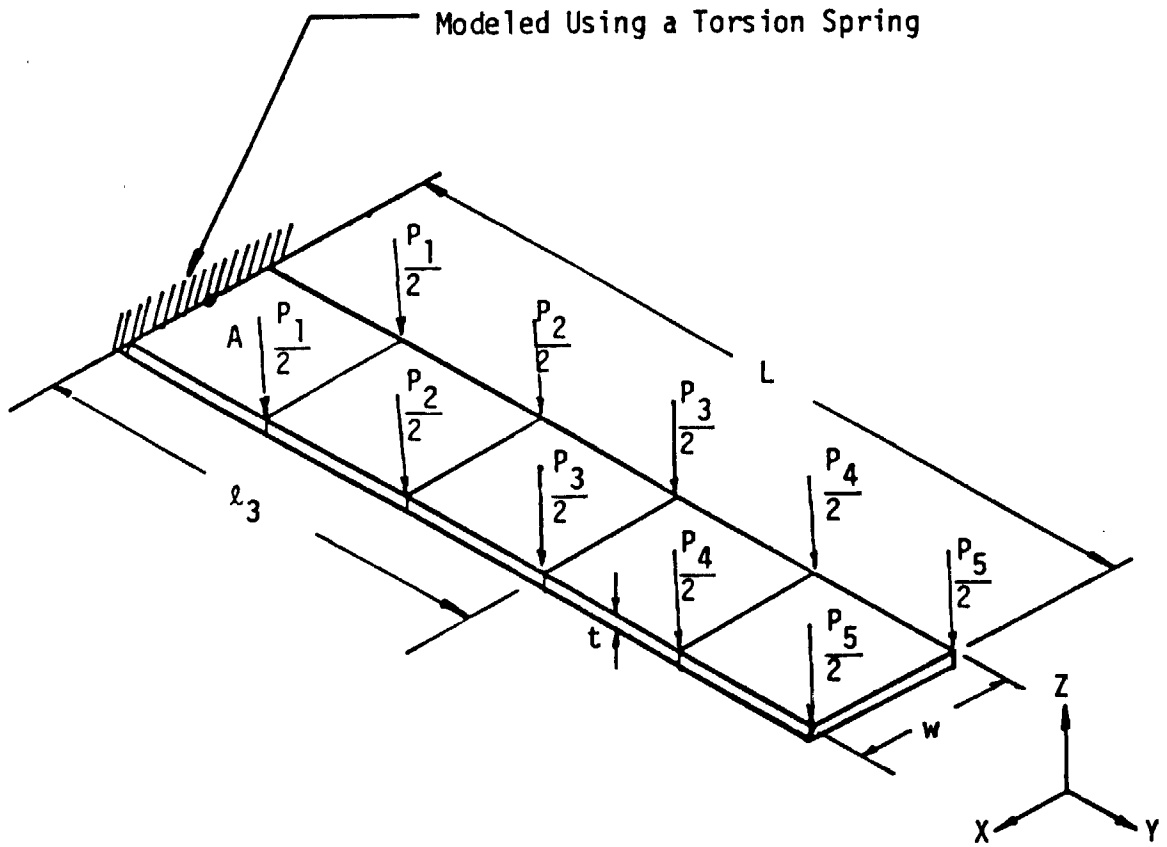


Fig. 2.17 Model Definition of a Cantilever Beam

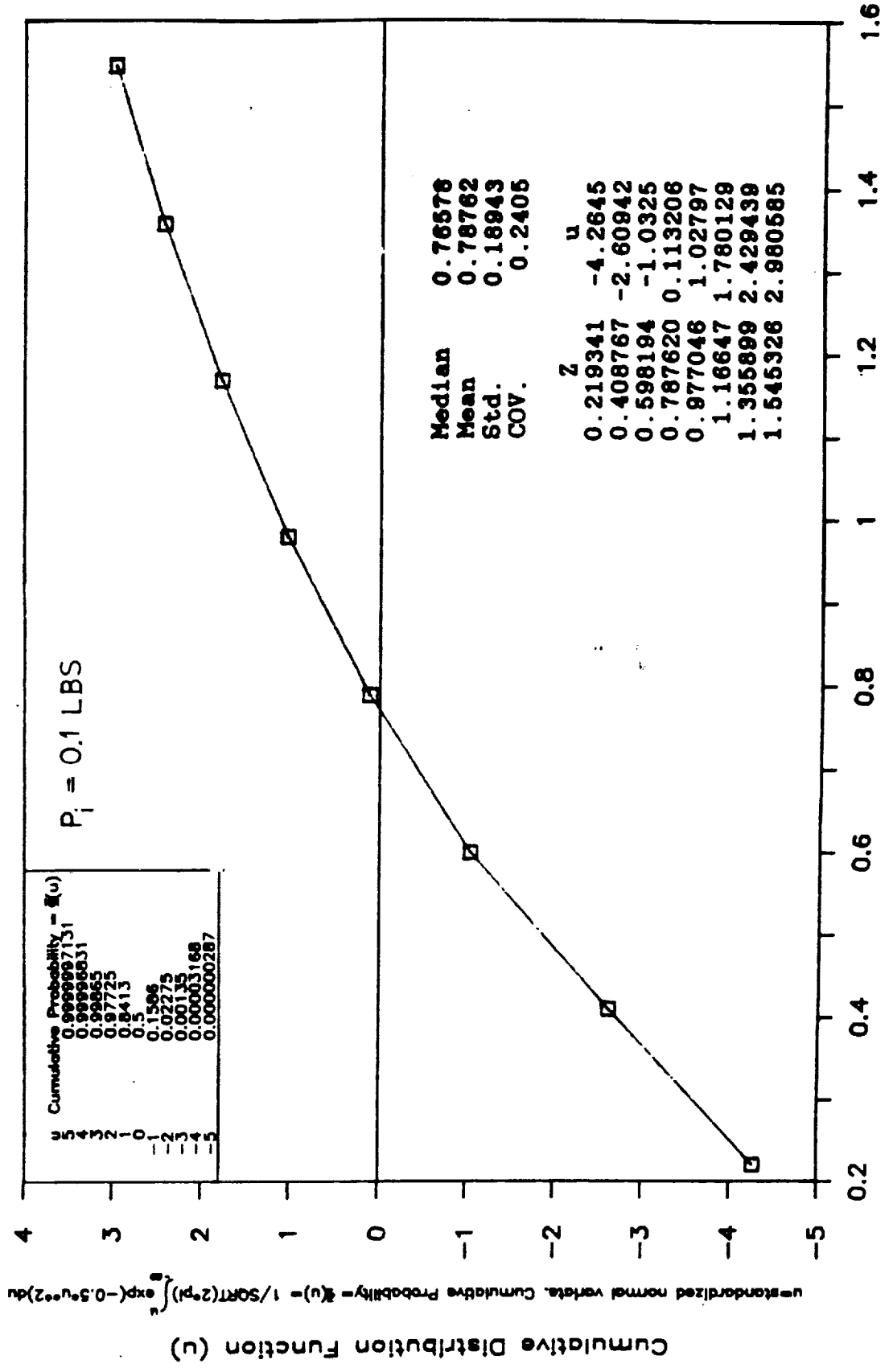


Fig. 2.18 Validation Test Case 2
(Exact CDF of Tip Displacement)

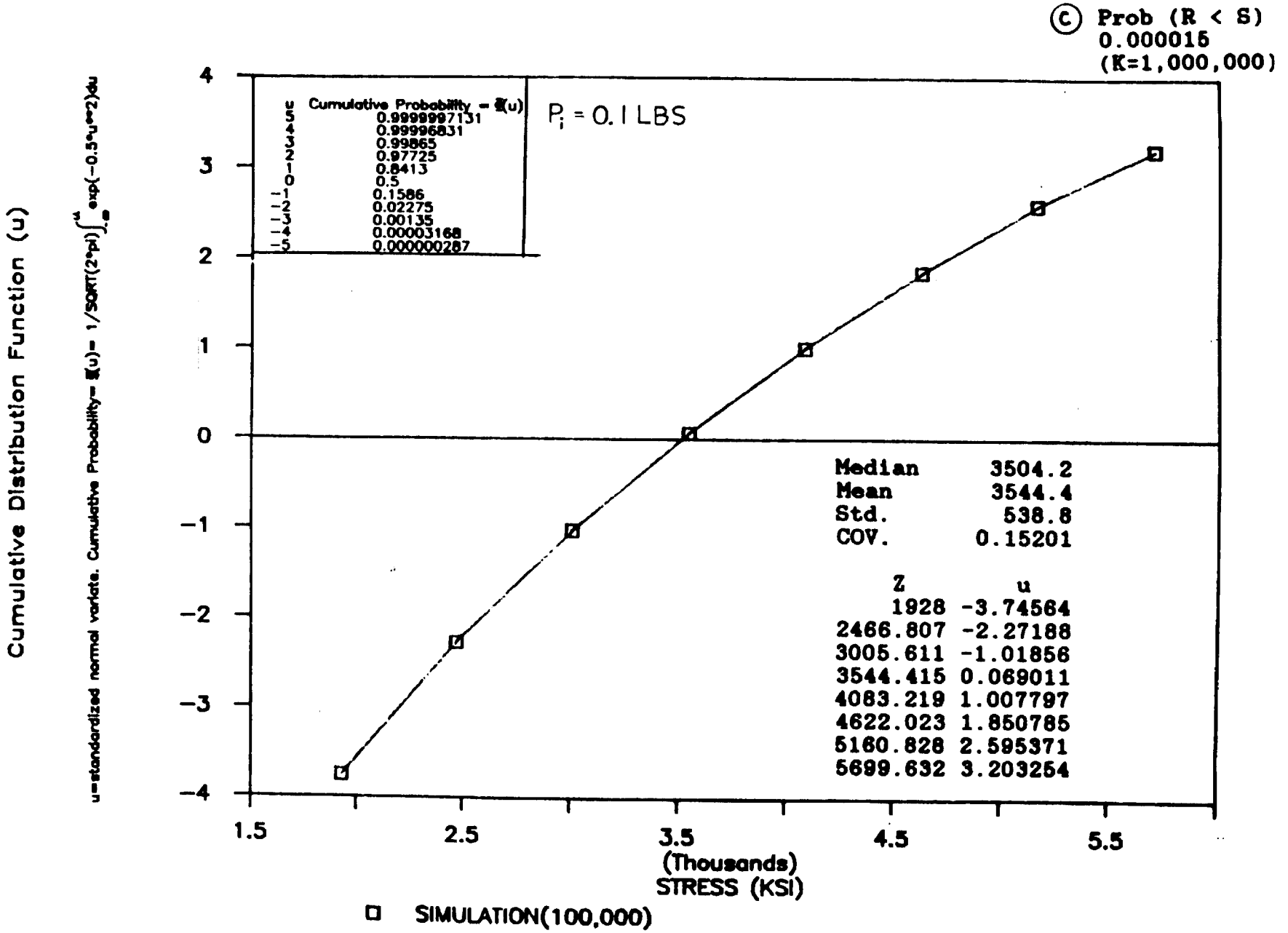


Fig. 2.19 Validation Test Case 2
(Exact CDF of Max. Stress)

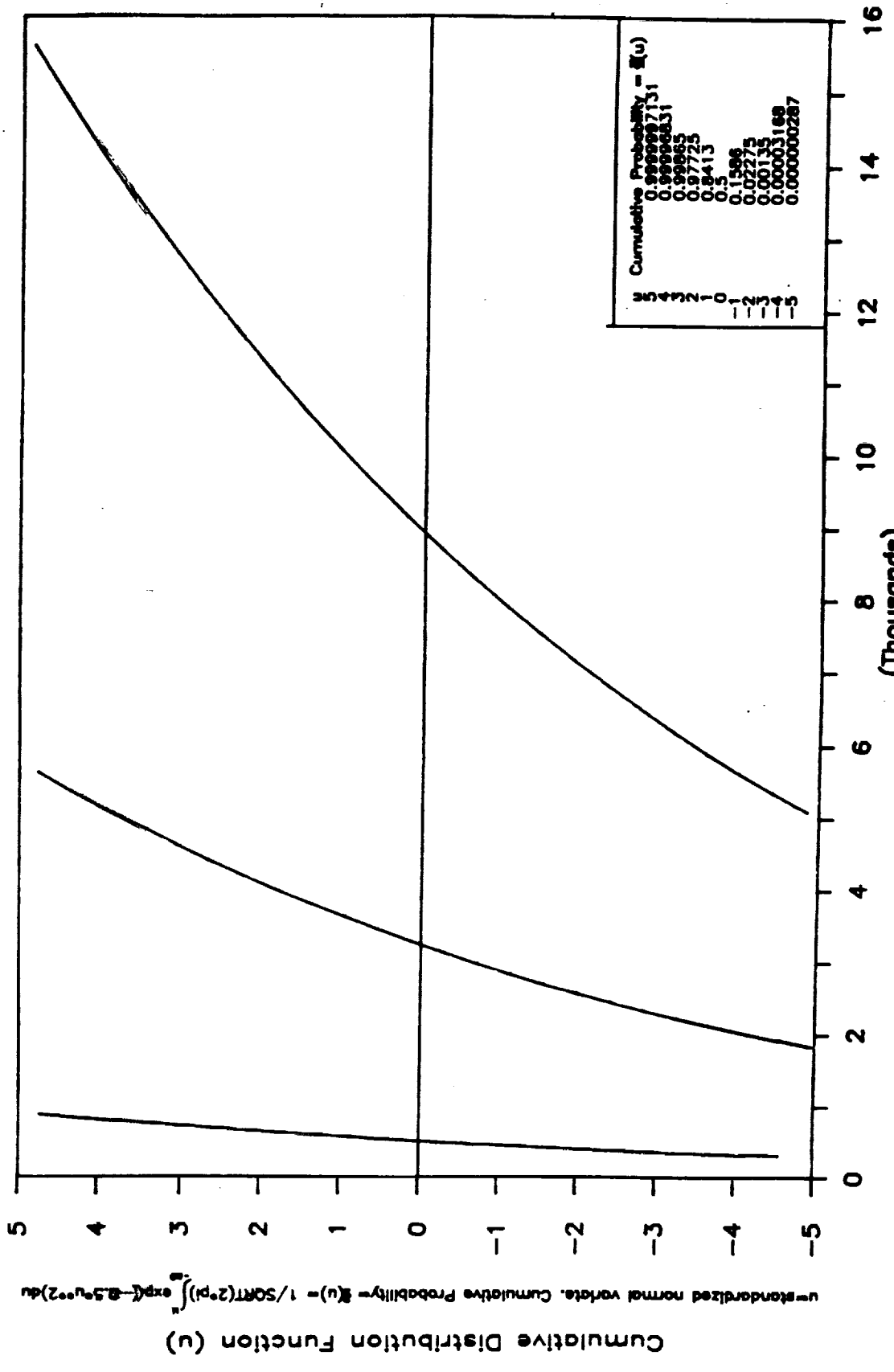


Fig. 2.20 Validation Test Case 3
(Exact CDF's of Frequencies (+/-X Dir.)

Table 2.4

Validation Case 3 Cantilever Beam (Natural Frequency)
X Dir. (Horizontal)

	Median	Mean	Std.
1st Mode	508.04	511.4438	59.28826
2nd Mode	3233	3254.661	377.2911
3rd Mode	8905.16	8964.824	1039.232

w(rad/sec)	u_1	u_2	w(rad/sec)	u_2	u_3
300	-4.56199		2567.145	-1.99717	
315	-4.13945		2695.502	-1.57464	
330.75	-3.71692		2830.277	-1.15211	
347.2875	-3.29439		2971.791	-0.72958	
364.6518	-2.87186		3120.380	-0.30705	
382.8844	-2.44933		3276.399	0.115481	
402.0286	-2.02680		3440.219	0.538013	
422.1301	-1.60426		3612.230	0.960544	
443.2366	-1.18173		3792.842	1.383076	
465.3984	-0.75920		3982.484	1.805608	
488.6683	-0.33667		4181.608	2.228140	
513.1018	0.085857		4390.689	2.650671	
538.7568	0.508389		4610.223	3.073203	
565.6947	0.930921		4840.734	3.495735	
593.9794	1.353453		5082.771	3.918267	-4.85640
623.6784	1.775984		5336.910	4.340798	-4.43387
654.8623	2.198516		5603.755	4.763330	-4.01134
687.6054	2.621048		5883.943		-3.58881
721.9857	3.043580		6178.140		-3.16627
758.0850	3.466111		6487.047		-2.74374
795.9893	3.888643		6811.400		-2.32121
835.7887	4.311175		7151.970		-1.89868
877.5782	4.733707		7509.568		-1.47615
1824.422		-4.95489	7885.047		-1.05362
1915.643		-4.53236	8279.299		-0.63108
2011.425		-4.10983	8693.264		-0.20855
2111.996		-3.68730	9127.927		0.213974
2217.596		-3.26477	9584.324		0.636505
2328.476		-2.84224	10063.54		1.059037
2444.900		-2.41970	10566.71		1.481569
			11095.05		1.904101
			11649.80		2.326632
			12232.29		2.749164
			12843.91		3.171696
			13486.10		3.594228
			14160.41		4.016759
			14868.43		4.439291
			15611.85		4.861823

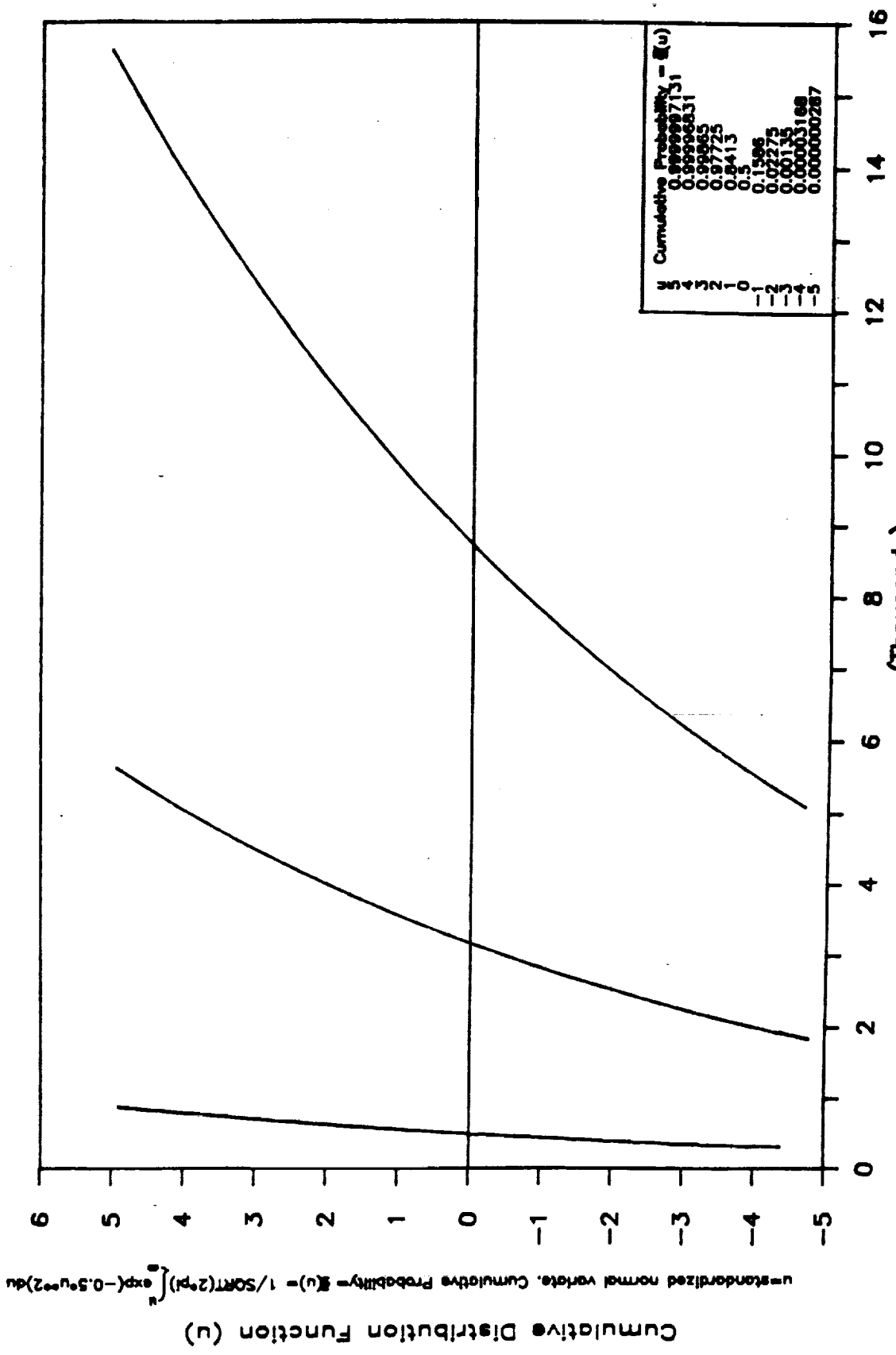


Fig. 2.21 Validation Test Case 3
(Exact CDF's of Frequencies (+/- Z Dir.)

Table 2.5

Validation Case 3 Cantilever Beam (Natural Frequency)
Z Dir. (Horizontal)

	Median	Mean	Std
1st Mode	497.9	501.2359	58.10493
2nd Mode	3168.5	3189.728	369.7639
3rd Mode	8727.4	8785.873	1018.487

w(rad/sec)	u_1	u_2	w(rad/sec)	u_2	u_3
300	-4.38739		2830.277	-0.97759	
315	-3.96486		2971.791	-0.55506	
330.75	-3.54233		3120.380	-0.13252	
347.2875	-3.11979		3276.399	0.290003	
364.6518	-2.69726		3440.219	0.712534	
382.8844	-2.27473		3612.230	1.135066	
402.0286	-1.85220		3792.842	1.557598	
422.1301	-1.42967		3982.484	1.980130	
443.2366	-1.00714		4181.608	2.402661	
465.3984	-0.58460		4390.689	2.825193	
488.6683	-0.16207		4610.223	3.247725	
513.1018	0.260455		4840.734	3.670257	
538.7568	0.682986		5082.771	4.092788	-4.68178
565.6947	1.105518		5336.910	4.515320	-4.25925
593.9794	1.528050		5603.755	4.937852	-3.83672
623.6784	1.950582		5883.943		-3.41419
654.8623	2.373114		6178.140		-2.99166
687.6054	2.795645		6487.047		-2.56912
712.9857	3.218177		6811.400		-2.14659
758.0850	3.640709		7151.970		-1.72406
795.9893	4.063241		7509.568		-1.30153
835.7887	4.485772		7885.047		-0.87900
877.5782	4.908304		8279.299		-0.45647
1824.422		-4.78037	8693.264		-0.03393
1915.643		-4.35784	9127.927		0.388592
2011.425		-3.93531	9584.324		0.811124
2111.996		-3.51278	10063.54		1.233656
2217.596		-3.09025	10566.71		1.656187
2328.476		-2.66771	11095.05		2.078719
2444.900		-2.24518	11649.80		2.501251
2567.145		-1.82265	12232.29		2.923783
2695.502		-1.40012	12843.91		3.346314
			13486.10		3.768846
			14160.41		4.191378
			14868.43		4.613910
			15611.85		5.036441

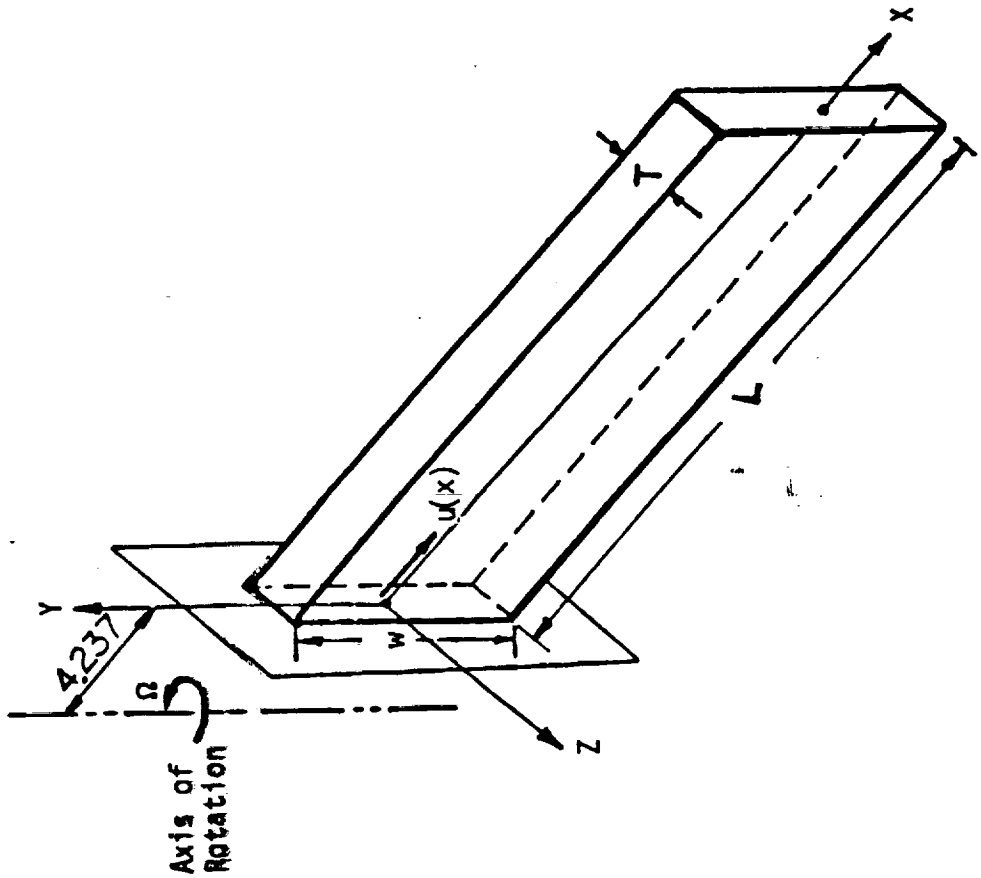


Fig. 2.22 Rotating Beam Geometry

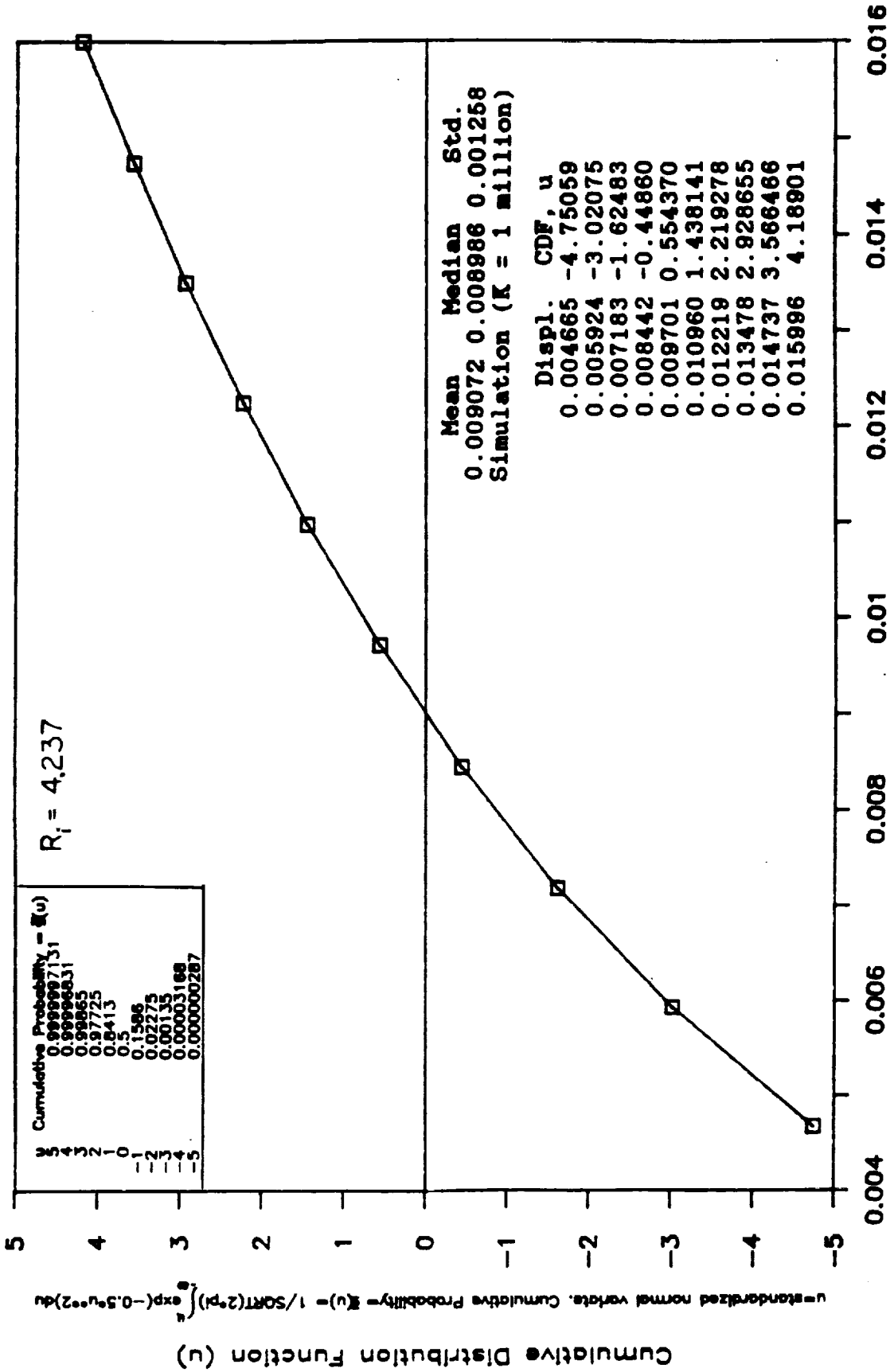


Fig. 2.23 Validation Test Cases 4 & 5
(Exact CDF of T1p Displacement)

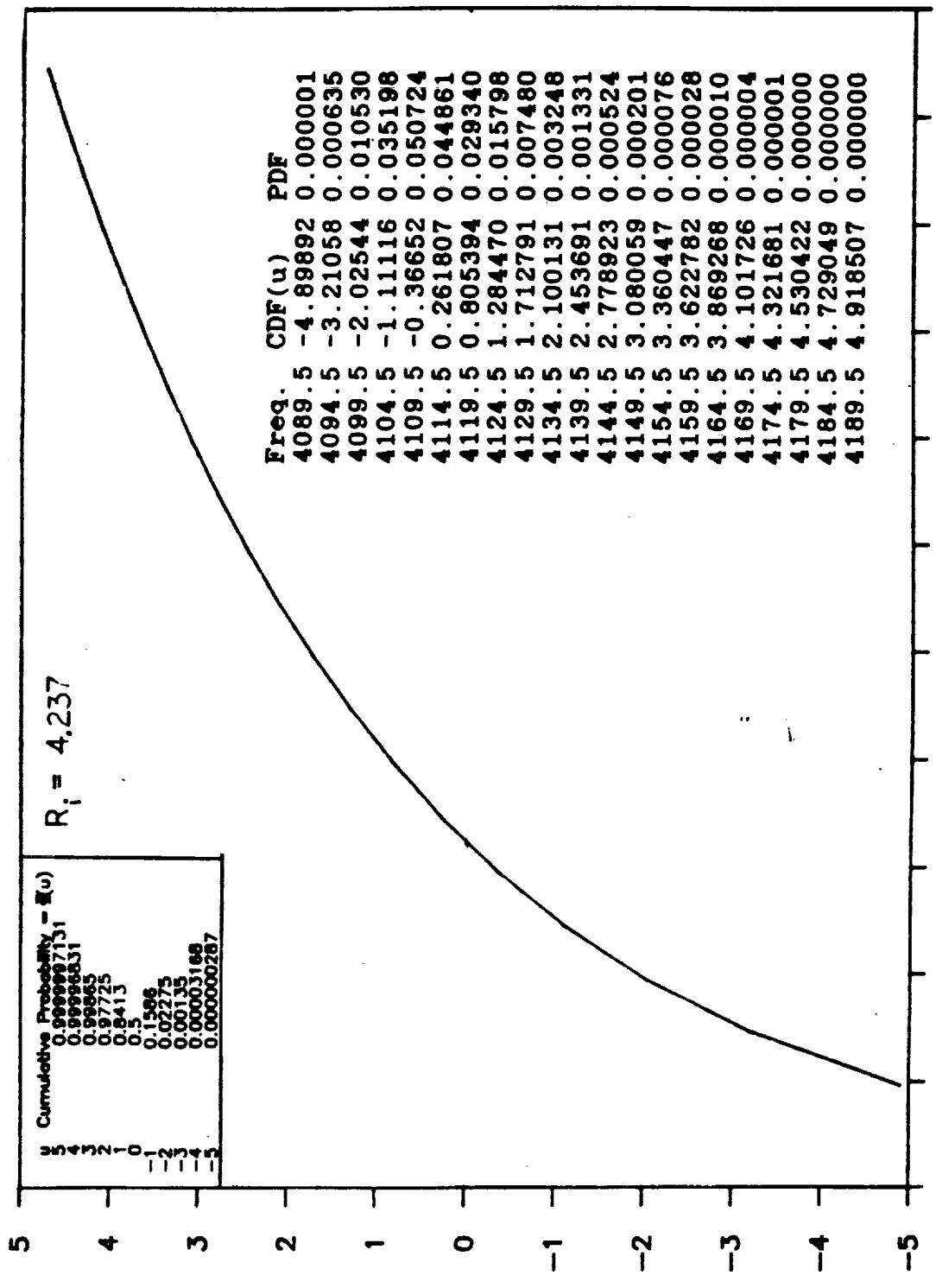


Fig. 2.24 Validation Test Cases 4 & 5
(Exact CDF of 1st Frequency)

solutions. To date these validation problems have been used to uncover several program bugs, to gain experiences for incorporating user-friendly interfaces, and to lead to new solution strategies.

The validation of problem 1 has raised an issue regarding the random variables data input structure. The type of finite element used in this problem is Timoshenko beam element (NESSUS element type 98). The random variables include thickness, t, and width, w, among others. The first-year NESSUS code defines the beam section using the area, A, and the area moment of inertias I_x and I_y. This format is not proper because A, I_x and I_y are correlated depending on the shape of the beam sections. Conseqently, the independent perturbations of w and t are impossible. To correct the dependency problems requires that the NESSUS/FEM code use "basic variables" w and t as input data. This strategy can be applied to other problems involving dependent variables.

Pending implementation of t and w as random variables, problem 1, with w and t as deterministic values, was used to validate other capabilities of the code. Modal frequencies, stress and displacement solutions were obtained and compared well with the analytical solutions. The perturbation solutions were not obtained, however, pending the code modification of the input structure.

For the validation problem 2, perturbation convergence instability has been observed for the width, w. In order to obtain a complete perturbation data base and to accelerate the validation process, w was temporarily treated as a deterministic value. The validation study of this slightly modified problem 2 has resulted in the successful integration of the NESSUS modules (PRE,FEM and FPI), using a successive linear approximation algorithm (Section 2.1.2.6). The study of the FPI iteration procedure for this problem has also led to a new and potentially more efficient solution strategy for estimating CDF (Section 2.1.2.7).

A validation problem not included in the first-year plan is a simple model simulating a turbine blade. The goal is to validate the capability of the code to treat the material axes as random variables. The model consists of four solid elements (NESSUS element type 75). The material has anisotropic property with one material axis modeled as a random variable. Perturbation results for the first two modal frequencies were obtained to estimate the CDFs using the NESSUS/FPI. Analytical solutions for this test

case is not available. However, the validation results were judged reasonable based on the information of the resulting means and standard deviations of the natural frequencies. A data input limitation was identified in that the material property (D) matrix defined in the FEM input data is deterministic. That is, the material properties such as the Young's modulus and the Poisson's ratio cannot be defined as random variables. It appears that code modification is necessary to solve the problem.

MARC has now completed the perturbation analysis for the validation problems 1, 3 and 5. New CDF estimation procedure (Section 2.1.2.7) will be used to continue the validation of the NESSUS modules and the solution procedure.

2.1.3 NESSUS/EXPERT Development

2.1.3.1 Approach

The goal of the expert user interface is to provide a flexible, user-friendly interface to the NESSUS/FEM and NESSUS/FPI codes. This interface will serve not only as an enhanced, on-line, automated user's manual for these codes, but it will also act as an expert aid in generating a data deck for a problem, especially the probabilistic information needed to solve a problem using NESSUS. Emphasis has been placed on minimizing the detailed knowledge that a user must have of NESSUS, allowing him/her to provide the information about a particular problem in as natural a way as possible and having the the expert user interface generate the actual data deck required.

To this end, an expert system called NESSUS/EXPERT is under development in parallel with the development of the NESSUS code itself. The system will consist of two major components, the interface to NESSUS/FEM and the interface to NESSUS/FPI. The interface to NESSUS/FEM is to contain essentially all of the knowledge about the use of NESSUS provided in the user's manual. It will also contain any clarification and other specifics about the use of the code known to those who developed the code and those who have tested it. It will also contain knowledge about generating probabilistic information about the problem from general descriptions. The interface to NESSUS/FPI will contain knowledge on how to analyze and interpret the results of a run, thus aiding the user in deciding what to do next.

Most such knowledge is embodied in the form of rules-of-thumb that provide methods for calculating specific values needed by NESSUS given general information about the problem, that provide hints about how best to use the system, and that indicate what is useful and important in the output of a run. Thus, the problem fits, in a fairly straightforward manner, the production rule knowledge representation method. This is convenient since most existing expert system building aids are rule-based and this is the best understood method of the AI technologies. Thus, the approach is to design and implement two rule-based expert systems to act as an intelligent front and back end to the NESSUS code.

2.1.3.2 LISP/OPS5 Environment

The programming language selected for initial development of NESSUS/EXPERT is OPS5. OPS5 is an expert system building software facility that allows a programmer to write production rules directly as code. The version of OPS5 being used in NESSUS/EXPERT is public domain and available free from Carnegie-Mellon University. It runs under Franz Lisp Unix on a DEC VAX. SwRI has recently ported this version of OPS5 to DEC Common Lisp so that it now runs under DEC VMS and on the Sun Workstation under Sun Common Lisp.

The entire NESSUS/EXPERT system will be coded intially using OPS5. The advantage of such a tool is that it offers a much higher level of productivity for the programmer because the knowledge can, to some extent, be encoded directly into OPS5 code. It also produces a much more readable and maintainable computer program. Though there are many other more elaborate, and more expensive, methods and tools for creating expert systems, the production rule technology embodied in OPS5 is sufficient for this task.

The major drawback of using a tool such as OPS5 for this application is its dependence on the Lisp environment. OPS5 is an interpreter coded in Lisp and, therefore, requires Lisp in order to run. Lisp does not currently provide an easy interface to FORTRAN on the DEC VAX. Thus, in the case where a data deck is produced for the pre-processor, the pre-processor cannot be invoked directly from NESSUS/EXPERT. Instead, the user must leave NESSUS/EXPERT, run the FORTRAN-based pre-processor, and then return to NESSUS/EXPERT where the resulting file can be read in and the process of developing a data deck for NESSUS can continue.

A Lisp-based system is being used because there are currently only about three expert system building tools available written in FORTRAN. Their functionality is limited and the resulting code is not all that readable because of the compromises made due to the FORTRAN language. A solution to this problem, as well as the requirement that all code for the PSAM project be delivered in FORTRAN, is to reimplement OPS5 in FORTRAN. The dependence on the Lisp environment would be removed and the interface to FORTRAN would be automatic. Another option would be to recode the entire NESSUS/EXPERT system in FORTRAN at the completion of this project. This is not desirable because all of the flexibility and maintainability acquired by using OPS5 will be lost in the translation. Therefore, for the moment NESSUS/EXPERT will remain in Lisp-based OPS5.

2.1.3.3 NESSUS/FEM Interface

Development of NESSUS/EXPERT has begun with the creation of an expert system for interfacing to NESSUS/FEM. Because the expert system developed must be an "expert" in how to use NESSUS/FEM, work has started by incorporating the knowledge contained in the MHOST User's Manual. Examination of the MHOST User's Manual supplied by MARC Analysis Resesarch Corporation has revealed a list of various types of knowledge that must be used when creating a data file for NESSUS/FEM that will run correctly for a specific problem. These include:

1. The required information for all problems (i.e., number of elements, connectivity of the nodes, etc.)

2. Interdependencies of options selected and data provided with other possible options and data (i.e., the number of elements provided under the model data must be less than or equal to the number provided under the parameter data, the *composite option under parameter data requires the *laminate option under the model data, etc.)

3. Incompatible selections of options/data (i.e., *frontalsolution option cannot be used with the *bandmatrix option)

4. Default options and values (i.e., *bandmatrix is the system default option, upper bound to the number of beam element crossings defaults to 1, etc.)

5. All available keywords and their "meanings"

6. Format of the parameters and data expected for each keyword, both for acquiring the needed information and for setting up the data file properly

7. Helpful hints concerning idiosyncrasies of the code (i.e., not recommended to use the option *stress, not recommended to use the option *displacement method for inelastic computations, etc.)

8. Helpful hints concerned with the "best" way to do something (i.e., for linear elastic analysis use the option *constitutive to avoid unnecessary computations, etc.)

9. What information about the problem can be inferred from other information. All but the last type of knowledge appears in the user's manual.

Of course, many of the first eight rule-types have been developed from talking with experts on the NESSUS code because the manual does not always provide all of the information necessary to run the code. However, it does provide an excellent place to start.

The overall design of the user interface maintains in spirit, anyway, the three step process used by the NESSUS code for developing a data deck for the FEM processor inputting the parameter data, the model data, and the incremental data, if needed. Input to the pre-processor is handled as a separate option in NESSUS/EXPERT. However, inputting the parameter data is not done immediately at the start of a session because many of its values can be inferred from the model data. Thus, the model data is input first, the necessary parameter values are determined by NESSUS/EXPERT and then the user is given a chance to enter whatever other parameter data he/she deems necessary. Each of the three steps consists of the following rule-sets:

- Rules to guide the questioning for required information and to check its correctness
- Rules to handle the optional, keyword input and to check its correctness
- Rules to handle a HELP facility
- Rules to output the data to the screen so that the user can verify the data
- Rules to check the completeness and consistency of the provided data
- Rules to write the data to a file in the proper format

Each of these groups of rules will constitute a separate portion of the knowledge base that we will refer to as rule-sets. They will

contain the various types of knowledge discussed in the previous section.

Overall, NESSUS/EXPERT for NESSUS/FEM can currently be characterized as a menu-driven consultant. Some input may come from previously prepared files, while other input can be supplied by the user interactively at the terminal. The basic tasks accomplished during each of the major phases in the system are described below. A block diagram of the system corresponding to this description appears in Fig. 2.25.

1. Identify Problem: During this beginning step, the user is asked to specify the name of the output file to be created and the type of data deck to be created. This information is then used by NESSUS/EXPERT to determine what should be done next.

2. Define a Preprocessor Data Deck: If the type of data deck to be created is a pre-processor data deck then the system follows the left branch of the flow diagram in Fig. 2.25. Currently, NESSUS/EXPERT is set-up simply so that such information can be entered through a dialogue guided by the expert system so that everything that is needed is entered. Each data set must consist of five categories of information: 1) RANDOM, 2) SELECT, 3) POINTS, 4) MEANS, and 5) DEVIATIONS. NESSUS/EXPERT simply prompts the user to enter all of this information during the dialogue. The structure for consistency checking of the data before it is written to the file is available, but currently no rules have been implemented. Any number of pre-processor data decks can be created during a given session. At the end of the session, the data is written to the file specified initially so that it can then be used by the pre-processor.

3. Initial Dialogue: If, during the initial identification dialogue the user specified that a FEM data deck is to be created, then the right branch of the system flow diagram given in Fig. 2.25 is followed. The user is asked to provide some introductory information and to complete the minimum subset of model data categories which constitutes a valid data deck. This information is extracted through an initial dialgue with the user which at the moment is an unvarying sequence of questions for which the user must supply answers. This area of the code will eventually need significant expansion from the AI point-of-view. Currently it only contains a minimum amount of knowledge that was derived from the MHOST manual. Eventually, it will include more detailed expert knowledge that wil help to determine the categories that should be included in this minimal data set based on some general questioning of the user.

4. Input Model Data: Most of the topics for the model data section of the NESSUS data deck are selected by having the user specify a topic by number or name from a large list of available topics. These topics correspond to the keywords used in the NESSUS code. Once a topic is selected, NESSUS/EXPERT guides the user in inputting the required information associated with that topic either by hand of from an existing file. When input is completed, control is returned to the main model data menu. Respecification and alteration of data

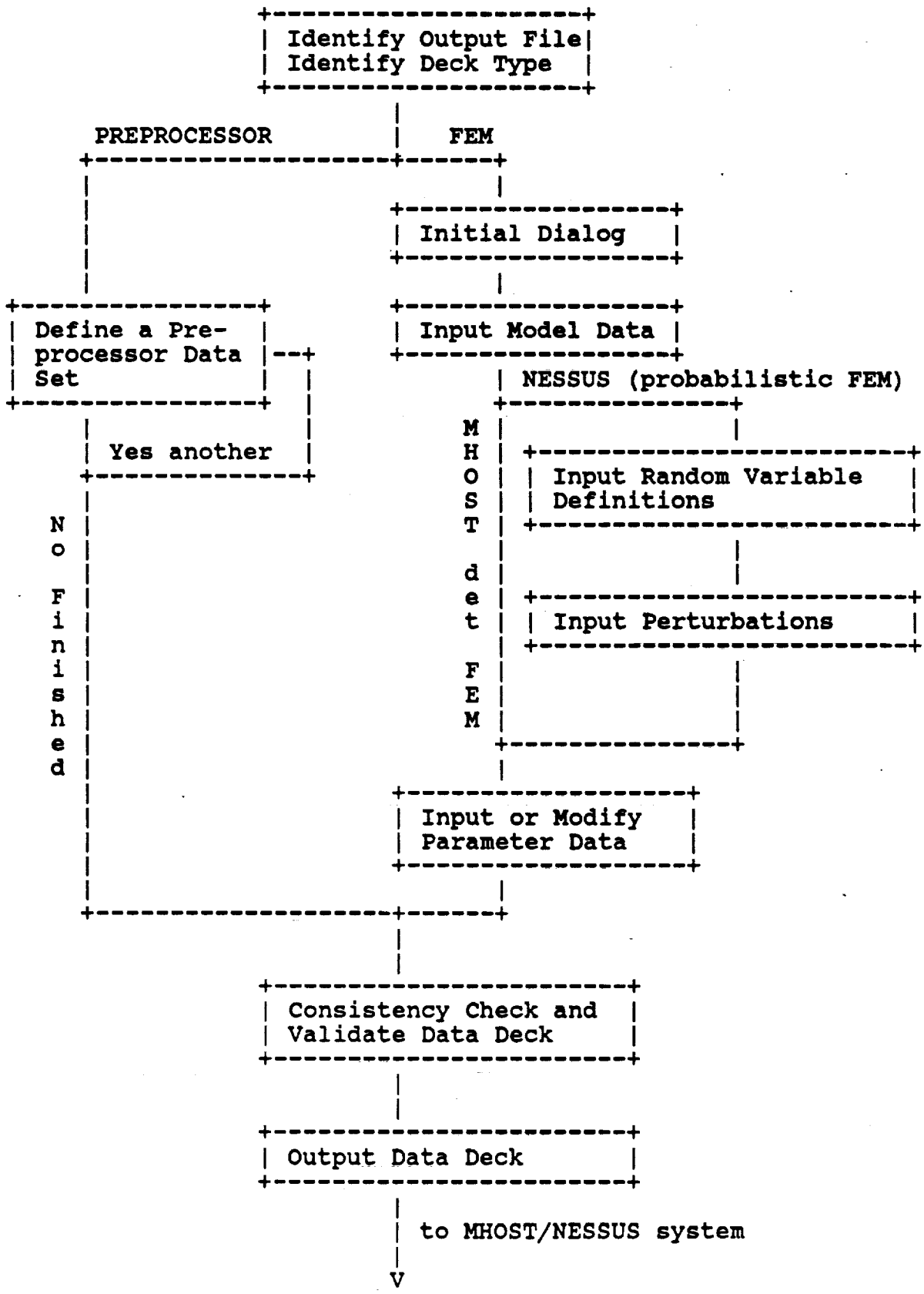


Fig. 2.25 Block Diagram of the NESSUS/EXPERT System

already provided is possible at all times. Also available is a help file on any of the topics that can be selected. Currently the knowledge that is contained in this section has come mostly from the user manual for MHOST. However, when ambiguity or inconsistency has appeared information has been acquired directly from the coders and testers of MHOST.

5. Input Random Variable Definitions and Perturbations: If a probabilistic FEM data deck is being prepared in the NESSUS/EXPERT session, the user will be asked to provide a set of random variables and perturbations once he/she has completed the model data section of the data deck. In overall style, the data entry in this section is handled in a manner highly consistent with previous sections of NESSUS/EXPERT. A certain set of keywords are needed, along with their corresponding piece of data. The system provides guidance in filling in the values associated with the the required keywords either manually or from a file. First, the definitions are simply asked for, then input of the perturbation information is guided by a parameter menu just like the model data section. As with all other sections of NESSUS/EXPERT, information can be corrected, deleted, or respecified at any time. This section currently embodies only the knowledge provided by Supplement II of the MHOST User's Manual. However, this section will require much more attention in terms of providing support to the user in the form of an intelligent aid for handling probabilistic geometric data in the coming year.

6. Consistency Checking and Validation of the Data Deck: Consistency checking of the completed data deck is one of the more important functions of NESSUS/EXPERT for it is here that much of the expert knowledge on how NESSUS works would be used to ensure a correct data deck. The goal of consistency checking is to determine whether the information in the completed data deck is consistent among all of the various categories. The rules encoded so far in NESSUS/EXPERT are, for the most part though sometimes very subtlely, contained in the MHOST User's Manual. Much of the knowledge has required clarification from either experts at SwRI or the original coders of the NESSUS system. When a problem is detected in the information provided in the data deck, the user is given a number of options for solving it, depending on the problem itself. Due to the power provided by a tool such as OPS5, all errors will be detected in a very straightforward manner and if another inconsistency is created by fixing a problem, this is detected as well. The knowledge encoded in the system so far has emphasized compatibility between the parameter and model data, between the BFGS and ITERATIONS data, between the CONSTITUTIVE and the WORKHARD data, between the random variable data for a particular topic of the model data and that model data topic, between the perturbation and random variable data, and within the WORKHARD data itself. This section will continue to be expanded for the duration of the project as this is where much of the intelligence of the NESSUS/EXPERT will reside.

7. Output Data Deck: Once the data deck has been completed and verified as being consistent (to the extent that is currently possible by NESSUS/EXPERT), the data deck is printed out to a file. It is done

in the following order: 1) the header records and deck title card, 2) the parameter data section, 3) the model data section, 4) the random variable section (if needed), and 5) the perturbations (if needed). The various sections are printed out in a suitable order (alphabetically or numerically as appropriate). This output goes to the file specified at the beginning of the session. Most of the basic structure of NESSUS/EXPERT exists now. What is left to do in many cases is to fill in the knowledge bases so that the coverage of NESSUS/EXPERT is complete. Other major additions left to be done are addressed in Section 3.1.3, Current Efforts on NESSUS/EXPERT.

2.1.3.4 Rule Structure

A production rule encodes knowledge about a problem in the form of IF-THEN statements also known as condition/action pairs. These production rules manipulate a set of data structures called objects. There can be an arbitrary number of these objects and each has associated with it a set of attributes and potential values for those attributes.

The generic form of an OPS5 production rule looks like the following:

```
(p ex-rule (object1 attribute1 value1 attribute2 nil) (object2
attribute3 <> value3) --> (make object4 attribute4 value4)
(modify 1 attribute2 value2) )
```

The letter "p" just inside the left parenthesis indicates the beginning of the production rule. The rule's name is "ex-rule". This allows the system to distinguish it uniquely from all other rules in the knoweldge base. The rest of the rule that occurs before the symbol "-->" is called the left-hand-side (LHS) of the rule. It contains two conditions. The first is that there exist an object1 with an attribute1 of value value1 and an attribute2 with no value. The second is that there exist an object2 whose value for attribute3 is not equal to value3. The portion of the rule following the "-->" symbol is called the right-hand-side (RHS) of the rule. It contains two actions. The first creates a new object, called object4 with attribute4 of value value4. The second modifies the first object listed in the LHS of the rule (object1) so that its attribute2 has value value2. Thus, if this rule were to become true, it would result in modifying the world of objects and attributes in that specified way.

In OPS5 such rules are used during processing by a

technique called forward chaining. This means that the rules are data-driven. Data in the world (i.e., the objects and their attributes) change. Such changes cause some of the LHS's of the rules in the knowledge base to become true. One rule from this set of true rules is selected through a method called "conflict resolution" as the appropriate rule to activate, or fire. Firing causes the actions on the RHS of the rule to be performed resulting in changes to the data in the world making a different set of production rules in the knowledge base true. This process of forward chaining continues until information is needed from the user or no more production rules can become true. If information is needed from the user, then this new information can modify the data in the world, thus resulting in continuing the chaining process. If no more rules are true, then processing stops.

One can represent fairly directly in OPS5 the knowledge needed for NESSUS/EXPERT, such as information concerning a certain piece of parameter data for NESSUS. For example, the object could simply be called parameter-data. Its attributes could include its name, its parameter-value names, and related model data names. The parameter data name's value could be ELEMENTS, its first parameter-value (element type) could be 7, and the related model data names would include ELEMENTS. A rule could then be devised that, based on the fact that the parameter data's name is ELEMENTS and its first parameter value is 7, can determine which pieces of model data are needed to run the problem correctly. The rule might look something like the following when converted into English: "IF there is an object called parameter-data, whose name is ELEMENTS and whose first parameter-value is 7, THEN the model data whose name is ELEMENT is also needed. This is a fairly obvious and simple rule, but they can become very complex, depending on what knowledge must be represented. The result of this rule is that if parameter data called ELEMENT exists in the data deck, then the corresponding model data called ELEMENT must also exist. This is a simple example ofhow consistency checking of the data deck can be done using OPS5 rules.

2.1.4 Verification Studies

2.1.4.1 Objectives of Verification Efforts

The basic objective of the verification effort is to apply the methods developed and implemented in NESSUS family of computer programs to the analysis of actual space propulsion system components. The

typical components to which the methods will be applied include a turbine blade, a high pressure duct, a lox post and a transfer tube duct liner. The verification efforts would cover a wide range of analysis options developed and implemented in NESSUS codes.

The knowledge gained in the verification efforts will be implemented in NESSUS expert system. The verification effort is broadly divided into simple verification and complex component verification analysis. Since NESSUS is in a state of continuous development during the contract, the simple verification studies are designed to meet the following objectives.

The simple verification models exercise the element types, the typical random variables, the range of perturbation of each random variable and various solution strategies for a particular component but on a simplistic model. These studies differ from validation studies by the fact that they are specifically targeted for each component analysis.

The results of the simple verification studies aid in establishing confidence in the code, identify its limitations in user interface, as well as analysis capabilities when applied to analysis of practical components. They also result in correcting element deficiencies and devise solution strategies that will be effective when analyzing full scale verification problems. The full scale verification problems on the other hand, if possible, are conducted on existing production finite element models and are typically expected to be much more computationally intensive requiring large main frame computing facility.

2.1.4.2 Scope of Verification Efforts

The space propulsion system components are subjected to environments with many random variables. Due to the difficulties in the instrumentation of high energy, high pressure and temperature systems, many variables are not well-characterized. Nevertheless, many components are subjected to severe environments. The current design philosophy is to analyze and design the components based on worst conditions using state-of-the-art deterministic analysis methods. The environments and conditions under which many space propulsion system components operate lead to structural analysis in the non-linear analysis domain. These structural analysis non-linearities can be due to material property or due to geometric changes or due to contact boundary conditions. Detailed discussion of the environments and

deterministic analysis techniques to which some typical space propulsion system components such as turbine blades, lox posts, transfer tubes, high pressure ducts, nozzle feed lines, and main combustion chamber walls are subjected to were described in detail in the first year report.

The composite loads spectra contract and probabilistic structural analysis contract are bold and challenging attempts to extend advanced deterministic structural analysis methodologies into probabilistic structural analysis domain. Developments under the PSAM contract are implemented incrementally into the NESSUS program during the five year contract period with increasing levels of analysis sophistication each year. Due to scheduling constraints, all analysis options available in NESSUS can not be applied to every component. Thus, a strategy has been developed in which the component, the type of structural analysis, random variables and the area of emphasis are chosen to be consistent with code development. This has been achieved in a probabilistic structural analysis domain for each component consistent with primary deterministic analysis requirement for each component. The scope of the verification studies achieves these objectives for each component in the order listed below:

1. Turbine blade
2. High pressure discharge duct
3. Lox post
4. Transfer duct

2.1.4.3 Turbine Blade Component Random Variables

The high pressure fuel pump turbine blade has been chosen as the first component to be analyzed by NESSUS finite element code. The analysis options and random variables chosen are consistent with the state of program development. The random variables that will be exercised on turbine blade analysis are:

1. Material property variations and orientations.
2. Geometry changes.
3. Centrifugal loads.
4. Pressure loads.

5. Temperature loads.

The strategy of the treatment of the random variables are first presented followed by the results of the simple verification studies.

2.1.4.4 Material Property Variations

The most commonly used turbine blade materials are nickel-based super alloys. Directionally solidified Mar-M-246 (Hf) is used in space shuttle main engine high pressure turbopump turbines. The single crystal PW1480 material is being evaluated for future use in the engine. These materials are anisotropic in nature and exhibit strong directionally oriented properties. As an example, for the PW1480 material at room temperature, the elastic modulus in the 111 plane can be as much as 250% greater than the modulus in 001 plane (Fig. 2.26). Thus, any perturbation of material orientation affects the blade stiffness and thereby its static and dynamic response. The material orientation angle is one of the random variables chosen in probabilistic structural analysis of turbine blades. Treatment of material orientation angle in single crystal blades is easier when compared to Directionally Solidified (DS) material blades. This is because the DS blade material typically contain a random number of crystals in each blade, (usually from 3 to 10), the volume of which can be random, with each crystal having its own material axis orientations. The single crystal materials, on the other hand, contain only one crystal but the orientation angles can vary slightly along the length of the blade based on crystal growth direction.

A typical statistical data of the distribution of the primary material axis orientation to the stacking axis from a set of hundred blades as measured at the base of firtree is shown in Fig. 2.27. Statistical analysis of data indicates a normal cumulative distribution provides a reasonable good fit of data. However, since blades having a cone angle of greater than 10° are rejected, the cumulative distribution function for the accepted blades is a truncated one modified as shown in Fig. 2.27.

Perturbation of material orientation angles is achieved in NESSUS by designating the orientation angles as a random variable. The studies of the perturbation of material orientation angles and the behavior of the numerical algorithm is discussed later in the section.

The other factor that might be considered in the material property variations is the scatter in elastic constants themselves from

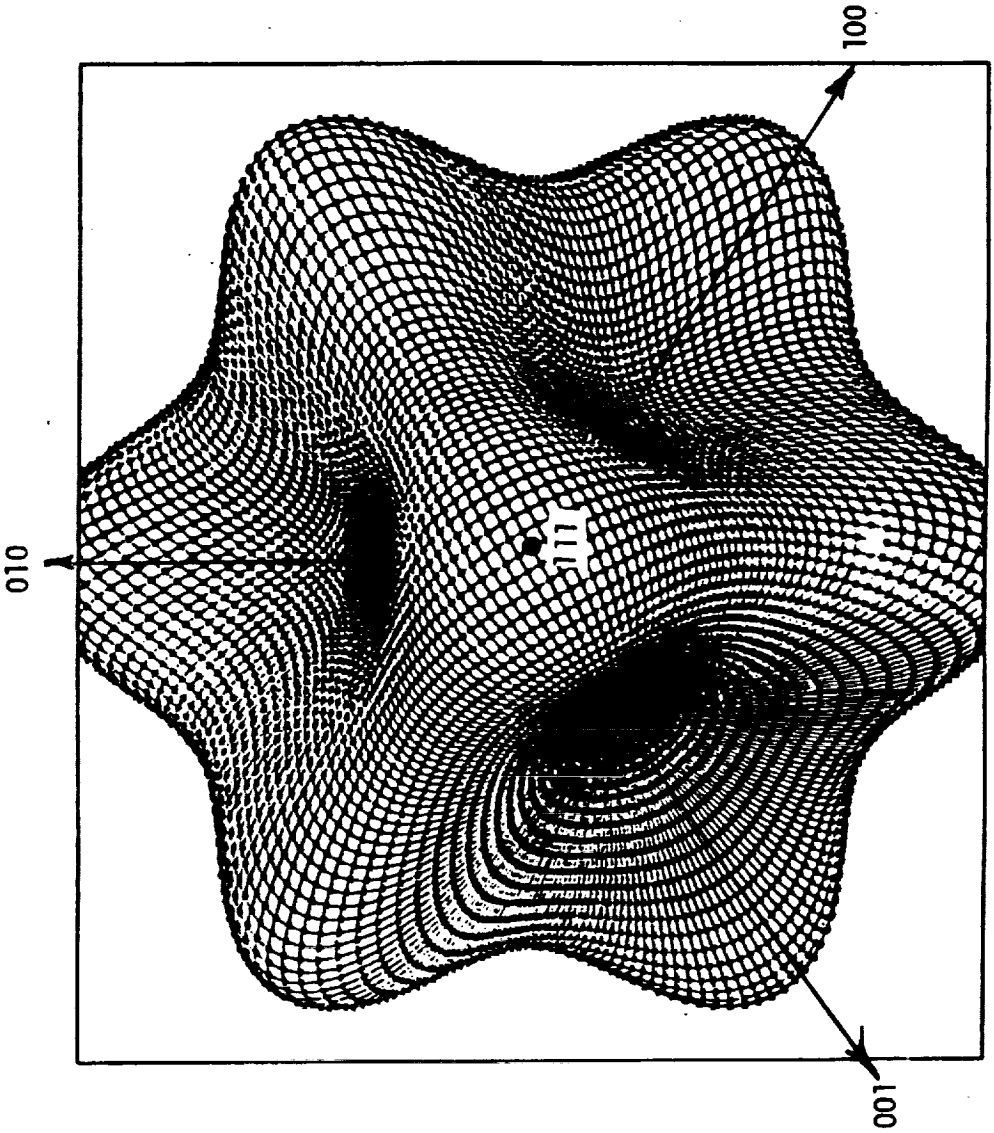


Fig. 2.26 Typical Elastic Modulus Variation From 001 Plane to 111 Plane for Nickel-Based Super Alloys

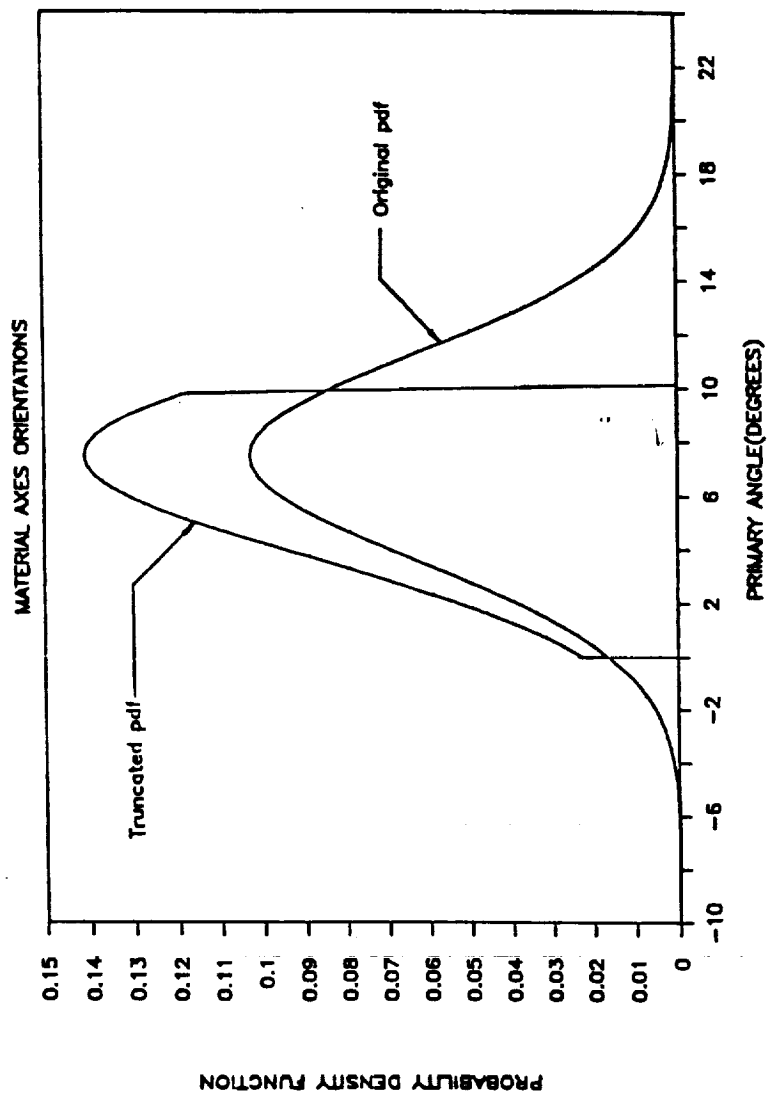


Fig. 2.27 Distribution of Primary Material Axis Orientation for a High Pressure Turbopump Turbine Blade

specimen to specimen at a given temperature. In general, insufficient amount of elastic properties data points exist at each temperature to do a good statistical analysis to accurately characterize the variations. However, this variation for elastic constants at a given temperature is small. A typical set of elastic constants for wide range of temperatures for PW1480 material is shown in Table 2.6.

Table 2.6

Elastic Constants for PW1480 Material as a Function of Temperature

	-400°F	70°F	1400°F	2000°F
E	19.96E6	18.38E6	14.75E6	11.0E6
G	20.50E6	18.63B6	15.27B6	12.82E6
n	0.376	0.386	0.395	0.416

The material property for anisotropic material is currently input to the code explicitly by specifying completing the material D matrix (s=De). However, for PW1480 material in the principal material orientations, a set elastic constants that can completely characterize the elastic response can be specified by E, n, and G. Thus, new features will be added to the code for specifying these constants (instead of the full D-matrix) and perturbations of them to calculate the response due to material property variations. The option of perturbing each coefficient of the full D-matrix is postponed to later releases of the code. The issue of building in rules in the NESSUS expert system to avoid material property perturbations that violate the laws of physics such as non-positive definiteness of the matrix will be addressed.

2.1.4.5 Geometry Changes

Because of the criticality of the component, every turbine blade that is used in an engine is subjected to quality inspection procedure for adherence to the design geometry. The blades that are used in space propulsion systems are typically short and compact, 0.5" to 3.0" in length when compared to turbine blades used in air breathing engines. The specified tolerance is a function of the manufacturing method. For cast blades, the tolerances are usually of the order of 0.005". Many turbine blades, including the kind used in the Space Shuttle Main Engine, are of cast

type with machined firtree which forms the mechanical attachment to the disk. The measured geometrical variations found in these blades generally fall in the category of relative twist of the blade (Fig. 2.28) and lateral shift of the profile within the tolerance envelope. This is presented in the form of center of gravity shift (x, y, coordinates) for a set of about seventy blades (Fig. 2.29). An analysis of the measured data indicates that a majority of geometrical variations from blade to blade occur when the firtrees are machined. The net effect of geometric variations introduced in this machining step is a rigid body shift of the airfoil, shank and platform relative to the stacking axis which runs at the center of firtree. Thus, the strategy that will be adopted for the perturbation of geometrical quantities for turbine blades will be the perturbation of nodel coordinates of the finite element model resulting from rigid body rotation about x, y and z axes rotations.

2.1.4.6 Centrifugal Load Variations

Centrifugal load is one of the primary loads on turbine blades. It contributes to a major share towards the mean stress level and thus plays a critical role in fatigue life calculations. The centrifugal load varies as the square of the turbine speed. The speed profile of high pressure fuel turbopump in SSME is shown in Fig. 2.30 which closely follows the engine thrust profile. An expanded trace of measured speed between 32000 to 36000 rpm from a pump signature test is shown in Fig. 2.31. Here, the power level was reduced 1% per three seconds of test.

Random speed oscillations can be seen about a mean from this data. Detailed study of test to test variations furnishes a good statistical database for this data. It is a level 1 type of probabilistic loading in that randomness of centrifugal load is spacially homogeneous for the finite element model. The engine balance models indicate that 2s speed variations at steady state power level for the SSME fuel pump is about 400 rpm out of 36600 rpm assuming a normal distribution. It is planned to use the actual processed test data from engine tests for the probabilistic structural analysis. The benefit of the results from the composite load spectra development contract will be utilized for all loads subject to their availability.

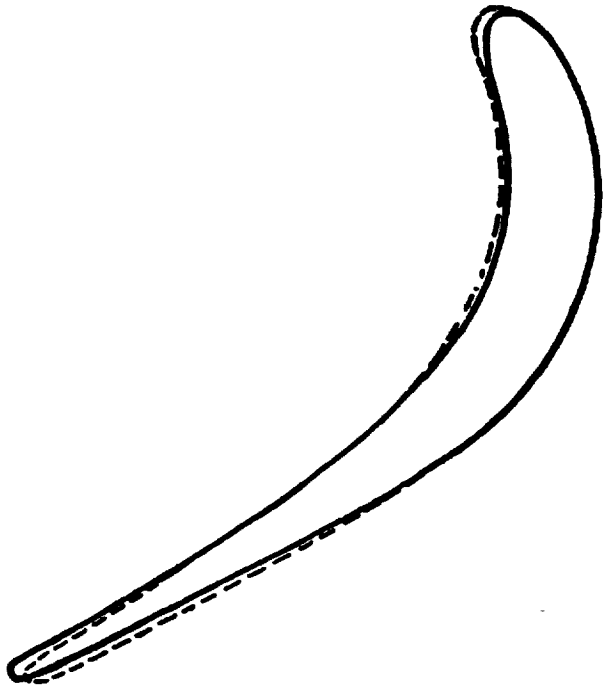
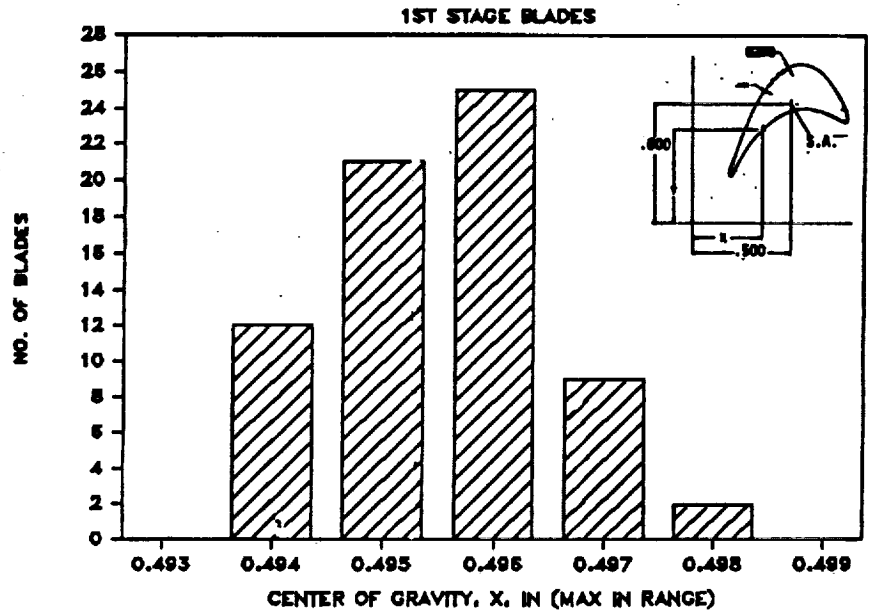


Fig. 2.28 Relative Twist of the Measured Blades from the Nominal



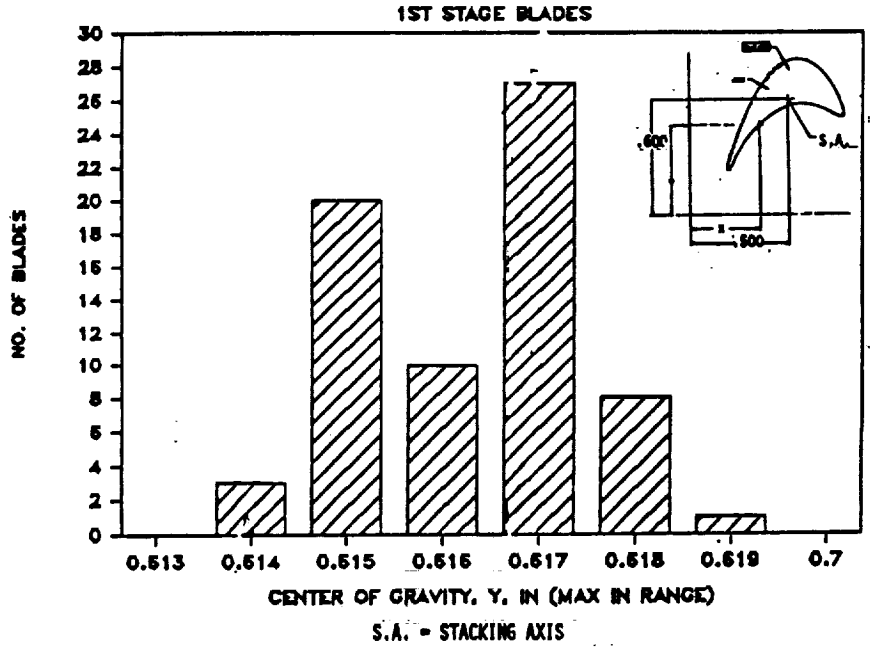


Fig. 2.29 Histogram of CG Shift of Measured Data for High Pressure Oxidizer Turbopump Blades

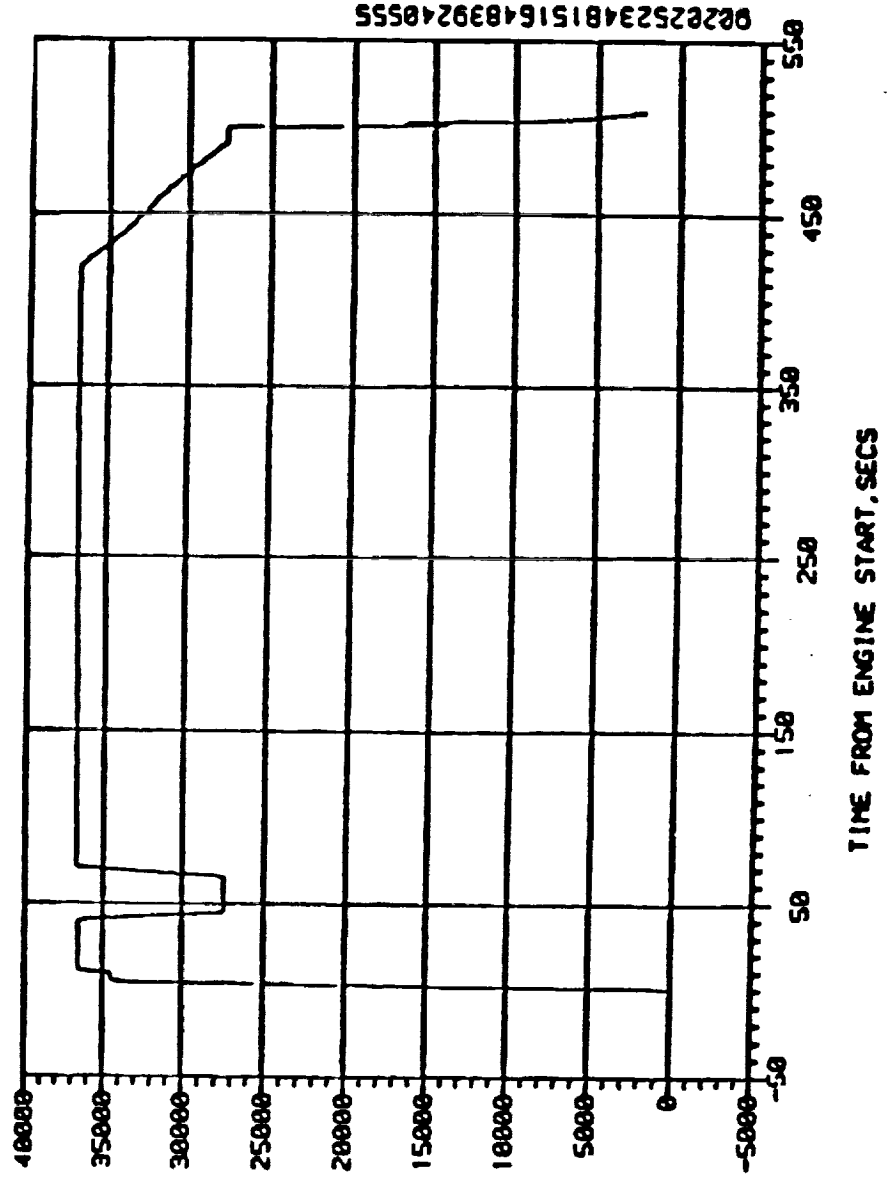


Fig. 2.30 A Typical Mission History Speed Profile of High Pressure Fuel Turbopump

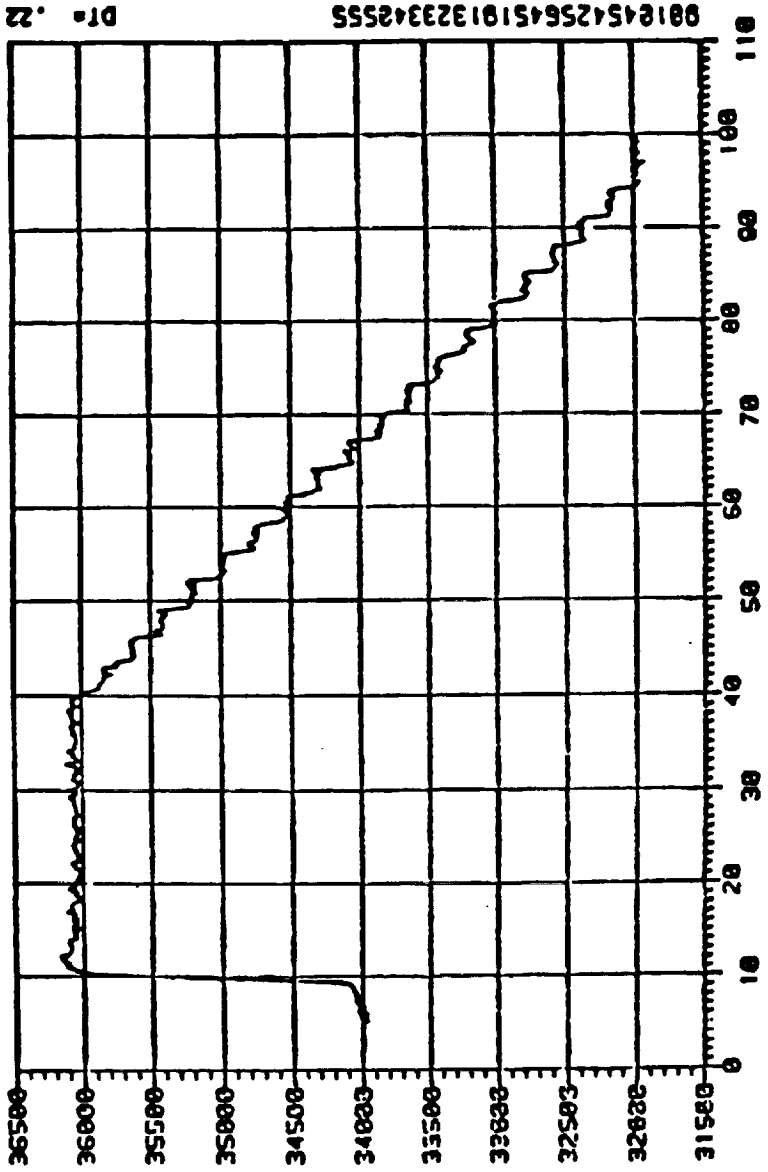


Fig. 2.31 Expanded Trace of Speed Profile From a Pump Signature Test

2.1.4.7 Steady-State Pressure Loads

The steady state pressure loads on turbine blades is a function of flow conditions at inlet and outlet of the turbine. Detailed measurements of turbine blade surface pressures and temperatures from actual engine tests are unavailable. There are a number of measurements such as preburner chamber pressure, downstream turbine discharge pressure and temperature (downstream of turnaround duct) and pump head raise measurements. There are a few measurements from instrumented turbopumps for temperature in the stators (nozzles) upstream of turbine blades. Thus, the fluctuation of static differential pressure on the turbine blade between pressure and suction faces will be a calculated quantity obtained from indirect measurements and theoretical engine models.

The type of stochastic modeling of pressure load on a turbine blade is closely related to the design features of the turbine. For the chosen high pressure fuel turbopump component, the design features are illustrated in Fig. 2.32. A notable feature is that this turbine has a secondary flow circuit for cooling the rotating hardware and includes cooling of the shank portion of the turbine blades. This cooling circuit affects the pressure in shank portion of the turbine blade. Thus, the pressure load on turbine blade will be treated as a random field, Level II type modeling. It is planned that the statistics of the differential pressure variation for the airfoil will be correlated through turbine torque variation. The shank pressure variations will be correlated to coolant pressure variations.

Typically, the pressure information will be available at three or four streamlines or cross sections which will be independent of the particular finite element model. The pressure at model node locations for a particular model will then have to be obtained through interpolator codes.

2.1.4.8 Blade Temperature Loads

The temperature loads plays a critical role in turbine blade analysis. For space propulsion systems of LOX/LH2 systems with staged combustion process, the range of temperatures can be very high in a duty cycle. For example, in SSME during one mission duty cycle, the blades will be a temperature range from 2200/R to 200/R. While it is virtually impossible to measure turbine blade temperatures in an actual engine, first stage stator (nozzle) temperature data from a few instrumented turbopumps is available. While temperature transients cause the worst case stresses when compared to

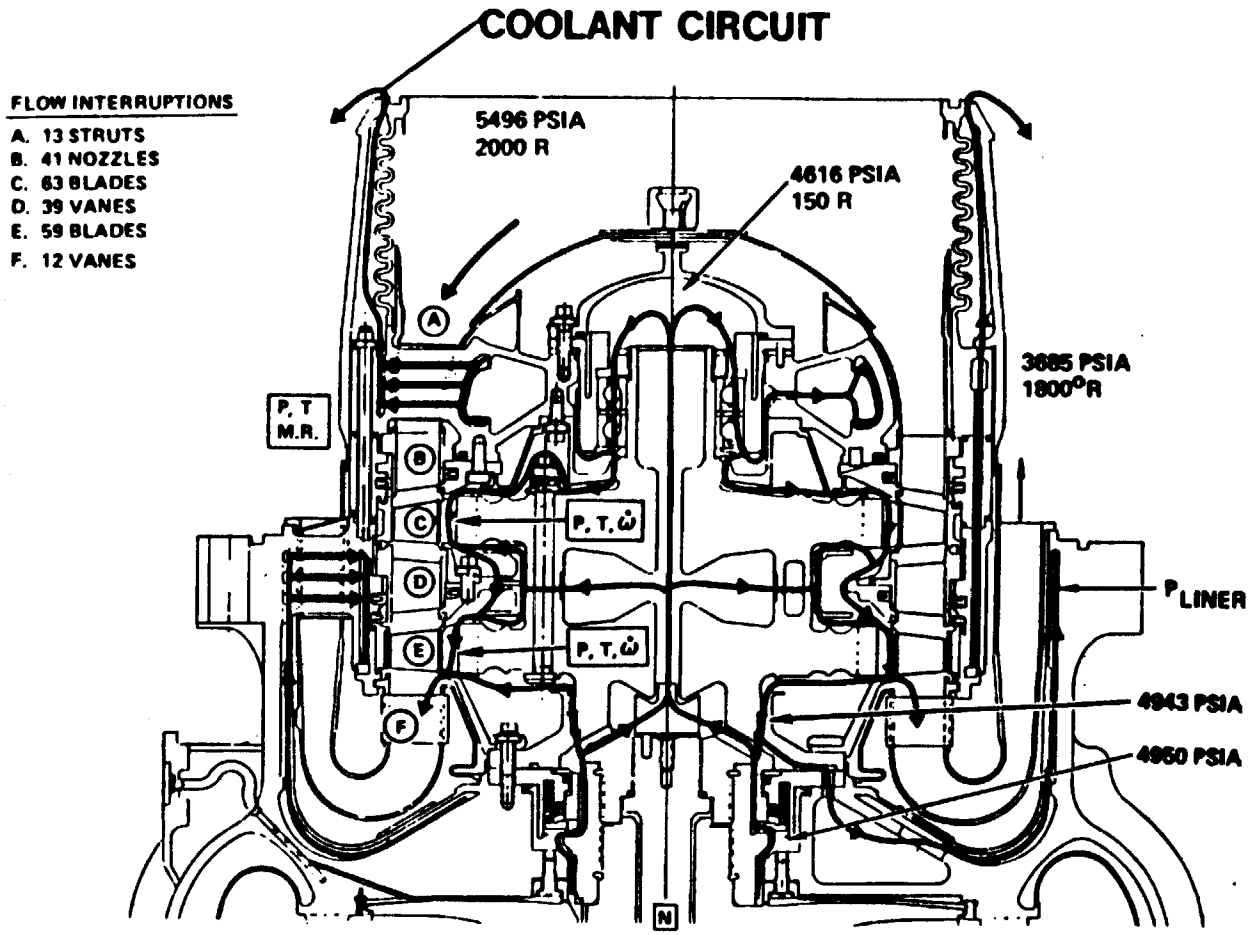


Fig. 2.32 High Pressure Fuel Turbopump Design Features

steady-state, the initial scope of the probabilistic structural analysis of turbine blade will be limited to steady-state loads. For this reason, the discussion is limited to solution strategies for the treatment of probabilistic temperature loads at steady-state.

Just as in the pressure case, the characteristics of the temperature random variable is closely related to the design features of the turbopump. For a high pressure fuel turbopump, the coolant flows around the shank. In actuality, the coolant and hot gas flows around the shank are very complicated. The hardware shows large variations in oxidation discoloration (which is a rough indication of temperature) from pump to pump, indicating that as the various seals wear they affect the flow circuit resistances and thereby temperature in the shank region. Thus, the developed probabilistic structural analysis methodology should be able to handle large local perturbations in temperature. On the other hand, the airfoil temperatures at steady state is essentially the hot gas temperature. Typically the shank area has a large thermal gradient when compared to the airfoil as shown in Fig. 2.33 and Fig. 2.34. The platform of the turbine blade itself is nearly isothermal at steady-state. Thus for the probabilistic structural analysis of HPFTP turbine blade, the temperature will be treated as a random field with varying statistical characteristics in airfoil, platform and shank. Thus, stochastic modeling of temperature is a Level II type modeling.

2.1.4.9 Deterministic Verification Solutions

Simple models, Fig. 2.35 through Fig. 2.37 comprised only of solid elements were exercised in NESSUS/FEM to understand and verify the performance of basic solid element as implemented in NESSUS. Several random variables were also exercised with typical range of perturbations that will be used in component verification studies. First, the deterministic results obtained from NESSUS are discussed, followed by perturbation analysis results. All the exercises were conducted on an anisotropic beam representative of PW1480 material properties at room temperature.

Considering centrifugal load first, model shown in Fig. 2.35 was exercised for both with one of the model axis as the axis of rotation as well as off axis rotation for hinged condition. The program results exactly match the theoretical calculated radial loads.

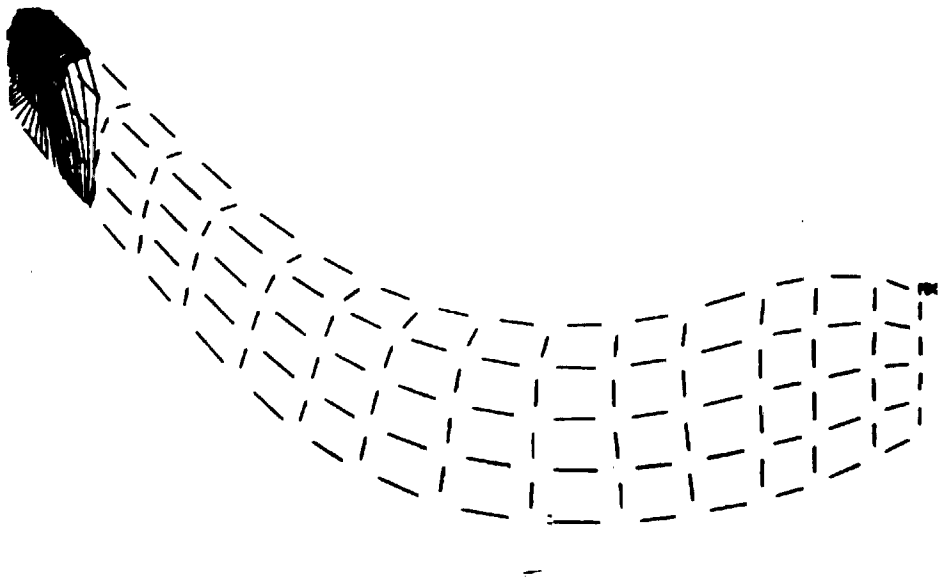


Fig. 2.33 Typical Temperature Gradient in the Shank Region for High Pressure Fuel Turbine Blade

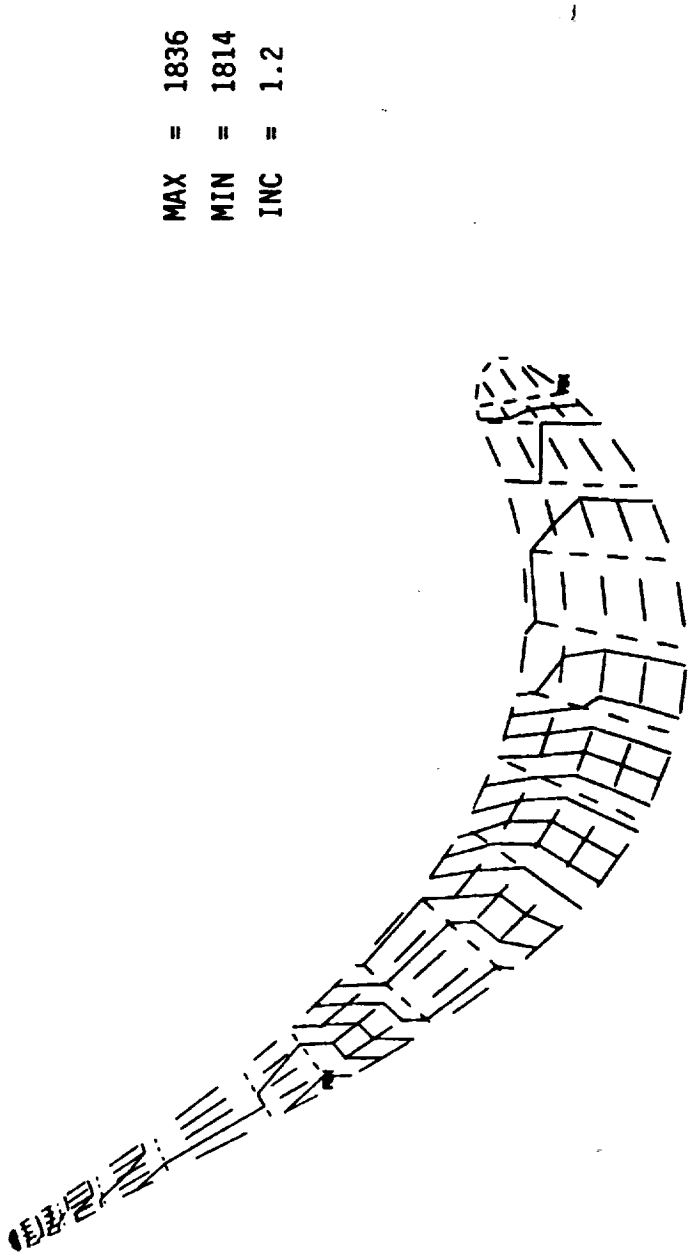


Fig. 2.34 Typical Temperature Gradient in the Airfoil Region for High Pressure Fuel Turbopump

C-2

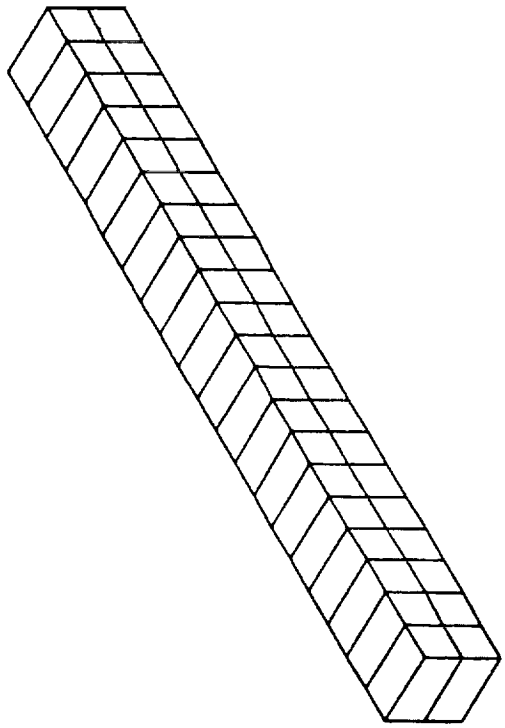


Fig. 2.35 Solid Element Beam Model A, 2 x 20 Elements

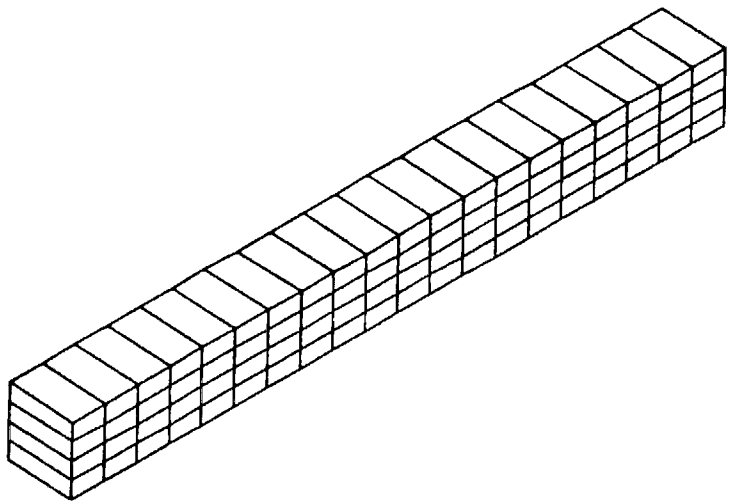


Fig. 2.36 Solid Element Beam Model B, 4 x 20 Elements

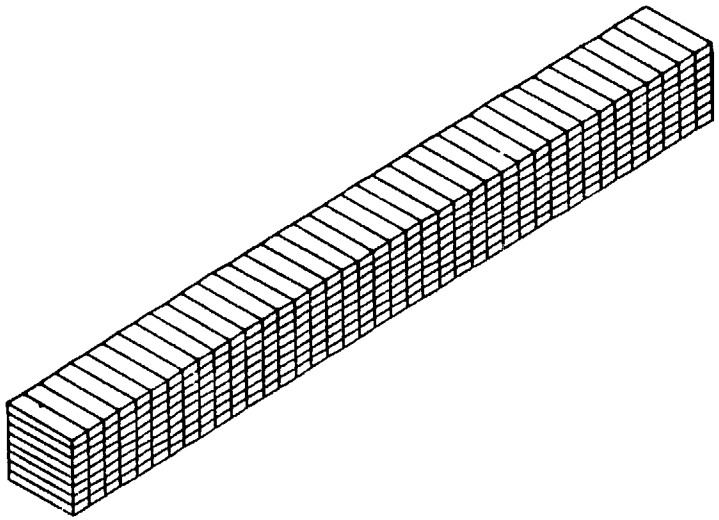


Fig. 2.37 Solid Element Beam Model C, 8 x 40 Elements

Considering the pressure loads next, the models in Fig. 2.35 - Fig. 2.37 were exercised for uniformly distributed pressure load and constant moment condition. The results of the uniformly distributed pressure load is presented in Table 2.7.

Table 2.7

Uniform Pressure Loading Results Cantilever Beam
FEM Results/Theory Ratio

	Simple Beam Theory	Model A	Model B	Model C
Tip Deflection	1.0	0.69	0.74	0.877
Fixed End Stresses	1.0	0.51	0.66	0.867

The basic solid elements as implemented currently in NESSUS is a strict eight-noded isoparametric element. It is known that these elements are stiff when they encounter pure bending situations and require a fine mesh to obtain good results. There are several approaches possible to improve the performance of this element. One of the well-known approaches is the introduction of additional modes such as $(1-r^2)$, $(1-s^2)$, $(1-t^2)$ for the 8-noded brick elements. While the introduction of these functions improves the performance dramatically for pure bending cases, they also violate compatibility and do not pass the patch test for arbitrary shaped quadrilaterals. Further, the performance deteriorates for arbitrary quadrilaterals. The problem of the patch test failure was subsequently cured by evaluating the contribution of the incompatible modes to the jocobian matrix at the centroid. It has been found that the resulting element gives superior performance to the original incompatible element.

The other approach to make the element flexible is through the use of reduced integration quadrature. The two concepts that are used are fully or uniformly reduced quadrature and selective reduced integration quadrature. Recent studies demonstrating the equivalence of a class of mixed models with reduced/selective integrated elements in linear elasticity has elevated the reduced integration approach from "tricks" to

legitimate methodology. However, the important considerations in the use of the methods are the insufficient rank of the matrix in the fully reduced method and the extension of the methodology to anisotropic cases in selective reduced method. The fully reduced quadrature is available in NESSUS without the hour-glass control. The fully reduced quadrature results in spurious modes, and therefore must be used with caution. For static analysis, computations using fully reduced integration scheme may be possible depending upon the boundary conditions providing stability to the problem. However, for transient dynamic analysis, hour-glass viscosity control to suppress the spurious modes is a necessity to obtain accurate results.

One of the principal deficiencies of the selective integration procedure or recently the B approach as normally implemented is that it is limited to isotropic case. For turbine blade applications, the material is anisotropic and the D-matrix is fully populated for general material orientation. The use of standard selective reduced integration schemes to anisotropic cases is ambiguous. Thus, it is desirable to implement extensions to selective integration schemes or to the B approach in the context of displacement formulation to cover anisotropic cases. The additional benefit of such a procedure would be its extension to nonlinear problems where tangent modulii always exhibit anisotropic character. Several temperature gradient solutions were also conducted on models Fig. 2.35 through Fig. 2.37 for the anisotropic material element. One of the notable features of the PW1480 material is that while its elastic properties exhibit strong directionally dependent properties, the coefficient of thermal expansion is nearly isotropic. The results of the temperature solution are presented in Table 2.8.

Table 2.8

Temperature Gradient Solution 1000° Through Thickness

	Simple Beam Theory	Model A	Model B	Model C
Tip Deflection (ratios)	1.0	1.17	1.02	1.12
Stress (absolute values)	0	128000	68000	33786

The maximum perturbations from 001 to 111 and vice-versa, were tested to check the convergence characteristics under maximum elastic property changes resulting from material orientation., In practice, material orientations are not allowed to differ more than ±10° from the primary direction. Thus, the Newton-Raphson method is expected to be adequate for material orientation perturbations for component verification. The same strategy should also be adequate for material property variations also as they are typically very small for single crystal blades.

Perturbation studies on geometrical changes are next addressed. The rigid body rotation about the base of the cantilever type geometric variations found in SSME turbine blades were earlier discussed. The greatest effect of this type of variation is in the contribution due the centrifugal load to the stresses due to change in the center of mass location. Two studies were conducted on the Model A (Fig. 2.35) where the geometrical perturbations were 1 degree and 10 degree rotational shift about the base of the rotating beam. for 1 degree perturbation, the default Newton-Raphson method converges for normal engineeringlimit of acceptable residual load errors. however, when the residual load vector is tightened to the order of 1E-5 of the total centrifugal load, the Newton-Raphson technique exhibits convergence and then divergence characteristics. However, when sealant iteration option is used, the algorithm exhibits uniform convergence and

converges to the tight tolerances (1E-5 of the total load) in three iterations. in actual turbine blades, variations of less than 1 degree tilt are expected. Thus, the available solution strategy appears adequate to handle geometric perturbations.

Due to the convergence behavior of Newton-Raphson technique for 1 degree perturbation, another case with a 10 degree perturbation was run. This case, the Newton-Raphson technique diverges from the start. The results are still under study. One fo the features of the test problem is the state of stress and centrifugal load in body fixed reference frame do not change due to perturbation. However, the global location of the body is different when measured from determinate reference frame after perturbation. The question of how large a perturbation the implemented solution strategies can tolerate will be studied further.

At the current state of development, NESSUS/FEM is applicable for linear analysis only. Thus, the perturbation of loads such as centrifugal and ressure loads amount to resolving the linear problem for a new load case with the old stiffness matrix. Irrespective of the magnitude of the perturbation of centrifugal and pressures, solutions converged in test cases in two iterations using Newton-Raphson method. Perturbation of loads and convergence have a greater bearing in the nonlinear analysis. The simple verification studies will continue to improve element and algorithm performances under a variety of conditions. Some of the improvements under development from the verification studies are described in the current efforts chapter of NESSUS/FEM. The results of the study will be used in component verification analysis of the turbine blade.

2.1.4.10 Perturbation Verification Studies

Perturbation verification studies were conducted on the model shown in Fig. 2.35. The random variables exercised to date include:

1. Material orientation angle

2. Nodel coordinates

3. Pressure

4. Centrifugal Load

The perturbation algorithm relies on established predictor-corrector methods used in nonlinear finite element analysis. There is no one iterative method that exists that exhibit ideal convergence characteristics as well as be cost effective in all situations. The solution strategy to be used is a function of the type of nonlinearity at hand. The methods that have been developed for nonlinear finite element analysis include full Newton, Quasi Newton, and Newton Raphson techniques. All the above techniques are available in NESSUS at a global level common to all perturbations within a run.

The logic for choosing the solution strategy should primarily depend on the rate of convergence and cost of the solution. A necessary condition for convergence for all the iterative methods is the exact calculation of residual load vector at each iteration. They all differ in the evaluation of predictor, the trial stiffness matrix used. In full Newton, the tangent stiffness is evaluated at every iteration. In the modified Newton-Raphson, the original stiffness matrix or the matrix at the start of the increment is used. In Quasi-Newton methods, the stiffness matrix is updated, but numerical strategies are used to reduce the amount of computations (update of stiffness matrix without inversion) than it would be if a full Newton method (requiring a full matrix inversion) was used. The initial exercises in the perturbation examples use the default Newton-Raphson method in the code. Other solution strategies were used only when divergence was encountered while using Newton-Raphson method.

The material angle perturbations are first addressed. The model (Fig. 2.35) was exercised for material axis variations in the presence of pure axial load. The objective of the studies were to test the convergence characteristics. One of the considerations was the study of the performance of the default Newton-Raphson method under perturbations that stiffen the structure. This can happen in turbine blade analysis when material orientation variations can result in a stiffer blade in the primary radial direction.

The study exercised the model in Fig. 2.34 with perturbations about the deterministic state resulting in stiffer or softer structure with varying magnitude. The results are summarized in Table 2.9.

Table 2.9

Material Orientation Angle Perturbation
Axial Load Results

Deterministic State	Amount of Perturbation About Deterministic State	Convergence	No. of Iterations for Residual Load Environment % of Applied Load 1	0.1	0.01
001	+ 10°	yes	4	8	16
001	To match 111 plane (36° + 45°)	no	-	-	-
111	+ 10°	yes	2	3	7
111	To match 001 plane (36° + 45°)	no	-	-	-

The maximum perturbations from 001 to 111 and vice-versa, were tested to check the convergence characteristics under maximum elastic property changes resulting from material orientation. In practice, material orientations are not allowed to differ more than +10° from the primary direction. Thus, the Newton-Raphson method is expected to be adequate for material orientation perturbations for component verification. The same strategy should also be adequate for material property variations also as they are typically very small for single crystal blades.

Perturbation studies on geometrical changes are next addressed. The rigid body rotation about the base of the cantilever type geometric variations found in SSME turbine blades were earlier discussed. The greatest effect of this type of variation is in the contribution due the centrifugal load to the stresses due to change in the center of mass location. Two studies were conducted on the Model A (Fig. 2.34) where the geometrical perturbations were 1 degree and 10 degree rotational shift about the base of the rotating beam. For 1 degree perturbation, the default Newton-Raphson method converges for normal engineering limit of acceptable residual load errors. However, when the residual load vector is tightened to the order

of IE-5 of the total centrifugal load, the Newton-Raphson technique exhibits convergence and then divergence characteristics. However, when sealant iteration option is used, the algorithm exhibits uniform convergence and converges to the tight tolerances (IE-5 of the total load) in three iterations. In actual turbine blades, variations of less than 1 degree tilt are expected. Thus, the available solution strategy appears adequate to handle geometric perturbations.

Due to the convergence behavior of Newton-Raphson technique for 1 degree perturbation, another case with a 10 degree perturbation was run. This case, the Newton-Raphson technique diverges from the start. The results are still under study. One of the features of the test problem is the state of stress and centrifugal load in body fixed reference frame do not change due to perturbation. However, the global location of the body is different when measured from determinate reference frame after perturbation. The question of how large a perturbation the implemented solution strategies can tolerate will be studies further.

At the current state of development, NESSUS/FEM is applicable for linear analysis only. Thus, the perturbation of loads such as centrifugal and pressure loads amount to resolving the linear problem for a new load case with the old stiffness matrix. Irrespective of the magnitude of the perturbation of centrifugal and pressures, solutions converged in test cases in two iterations using Newton-Raphson method. Perturbation of loads and convergence have a greater bearing in the nonlinear analysis. The simple verification studies will continue to improve element and algorithm performances under a variety of conditions. Some of the improvements under development from the verification studies are described in the current elements chapter of NESSUS/FEM. The results of the study will be used in component verification analysis of the turbine blade.

3.0 CURRENT EFFORT

3.1 NESSUS/FEM

Two different approaches have been proposed for the extension of the NESSUS perturbation algorithms to inelastic problems. The first approach calls for continuing the development within the displacement formulation used in the first year PFEM effort. Extension of the displacement formulation to inelastic analysis in NESSUS/FEM will require a major reorganization of the internal data structures within the code. The second approach calls for the adoption of a mixed iterative formulation, which would preserve the internal data structure of the present code. The development and implementation of appropriate perturbation algorithms for inelastic analysis will be started as soon as a decision is reached regarding the finite element formulation adopted for future PFEM development.

The development of a finite deformation kinematics algorithm for NESSUS is currently well underway. The adopted formulation utilizes an updated Lagrangian mesh description, with a constitutive relation based on the Green-Naghdi rate of Cauchy stress and rate of deformation. Although the development of nonlinear displacement and strain modeling capability is not required in NESSUS/FEM until FY88, MARC has taken advantage of the development of a similar capability for the MHOST code. The finite deformation algorithms being developed for the MHOST code will be added to the main development version of NESSUS/FEM in a very near future.

An enhanced continuum-based plate/shell element with surface node definition is currently under development at MARC. This element is envisioned as an eight-node brick with assumed strain modes based on the exact bending solution for an elastic isotropic material. The approach is expected to result in a non-locking element with enhanced bending behavior which can be distorted to a high aspect ratio ($h/L \simeq 1/10$) in order to model moderately thick plate and shell-like structures. An early version of this element for use in linear elastostatics should be available in time for the 2/1/87 code delivery.

A revised format to allow specification of surface pressures and edge tractions on a nodal basis will be developed and tested. Changes will be implemented to allow the degeneration of continuum-type elements to form triangles, wedges and tetrahedra. This will require changes to the strain

smoothing procedure available in NESSUS 1.1. The new smoothing procedure will then be tested for robustness.

The test cases proposed in the preliminary plan for validation of the NESSUS code are being exercised at MARC. In addition, MARC is in the process of compiling a standard list of test problems that will be used to exercise all versions of NESSUS shipped from MARC. These problems range in size and complexity from small one element tests to irregular element meshes of a few hundred degrees-of-freedom.

3.2 NESSUS/FPI

Testing of the new CDF estimation procedure (Section 2.1.2.7) and the validation of the NESSUS code is in progress. The exact solutions of the validation test problems have been obtained for the first five problems (Section 2.1.2.8). Solutions for the remaining problems will be obtained in the current year. These solutions will be used to compare results generated from the NESSUS/FPI. By using the perturbation data base generated by MARC (perturbation solutions are now available for the validation problems 1, 3 and 5), the new CDF estimation procedure will be used to continue the validation of the NESSUS modules and the solution procedure. The solutions will require additional runs of the deterministic FEM solutions and, if necessary, additional perturbations.

Effort in integrating the NESSUS/FPI with the NESSUS/PRE, NESSUS/FEM and NESSUS/EXPERT is in progress. The basic structure of the expert system code NESSUS/EXPERT is in place and operational. The emphasis during the next year will be to make the code easier to use by the engineer.

One of the difficulties identified in conducting probabilistic structural analysis on systems with a large number of random variables is developing a method of efficiently entering the random variables into the computer. For the analysts to enter a separate probabilistic data base would be time consuming and error prone. The approach being pursued is to use the existing data base for the structural model along with the NESSUS/EXPERT to query the user as to which variables are random. Distributional information and the degree of correlation will also be provided at this time. With this information, NESSUS/EXPERT can generate an input file for the FORTRAN code NESSUS/PRE.

The user will now have to exit NESSUS/EXPERT to run NESSUS/PRE. However, prior to exiting, NESSUS/EXPERT will save a data file of the

model and the random variables. NESSUS/PRE transforms a set of correlated random variables to a set of uncorrelated random variables using the eigenvector transformation method. This set of uncorrelated variables are saved in a file. Finally, the user will have to enter NESSUS/EXPERT again. NESSUS/EXPERT will retrieve the previously stored files and generate a complete NESSUS/FEM file which includes the structural model data, the random variables data and the perturbation settings.

3.3 NESSUS/EXPERT

Now that the basic structure and approach to NESUSS/EXPERT has been designed and implemented, emphasis is turning to an evaluation of this initial prototype to determine what is good and bad about it. Work will also proceed on extending the knowledge base to include knowledge of all keywords listed in the MHOST User's Manual. Finally, once the results of the prototype evaluation are completed and implemented, work will begin on handling the probabilistic data in a more natural and intelligent manner.

Extensive discussions between the experts on the use of NESSUS and the knowledge engineer implementing NESSUS/EXPERT have already begun. results so far indicate that some changes to the basic control structure need to be made in order to take advantage of some overlap in the use of certain data in different sections of the input data deck. The result will probably be a major change to the overall flow diagram given in Figure 2.25. However, because of the use of a very high level language such as OPS5, the necessary changes should not be difficult to make.

Work on enhancing the knowledge base will not proceed until the basic changes to the prototype flow of control have been made. The major source of knowledge will be the MHOST User's Manual and the knowledge will emphasize the sue of keywords. More held files and consistency checking rules will also be added as the project progresses. When the information in the User's Manual is incomplete or ambiguous, knowledge will be solicited from human experts on the use of MHOST.

Handling the input of the probabilistic data in a natural and intelligent way will require some research on what the best interface might be. Currently, the method of inputting of the data is the same as for the model data. The knowledge that this section of NESSUS/EXPERT contains is simply information about the keywords pulled mainly from the User's Manual. However,

this method is not very helpful or efficient for the user to work with when entering such information. Possible enhancements include providing some graphic aids that can illustrate various permutations on an element and some intelligence of probability as it relates to FEM so that NESSUS/EXPERT can make many of the decisions and perform many of the needed calculations itself, rather than making the user do them.

4.0 REFERENCES

[1] O.H. Burnside, et al., "Probabilistic Structural Analysis Methods (PSAM)," 1st Annual Report, NASA Contract NAS3-24389, Vol. I.

[2] O.H. Burnside, et al., "Probablilistic Structural Analysis Methods (PSAM)," 1st Annual Report, NASA Contract NAS3-24389, Vol. III, Section 5.

[3] A. Der Kiureghian and P.-L. Liu, "Structural Reliability Under Incomplete Probability Information," Report No. UB/SESM-85/01, Department of Civil Engineering, University of California, Berkeley.

[4] Y.-T. Wu, "Demonstration of a New, Fast Probability Integration Method for Reliability Analysis," Proceedings of Symposium on Probability Structural Design and Analysis, ASME, 1985.

[5] O.H. Burnside, et al., "Probabilistic Structural Analysis Methods (PSAM)," 1st Annual Report, NASA Contract NAS3-24389, Vol. I, Section 6.

[6] T.-L. Chang, "Investigation of the Performance of the Wu Algorithm for Computing Structural Reliability," M.S. Report, University of Arizona, 1985.

[7] P.H. Wirsching, "Performance of Wu/FPI Relative to Second Order Reliability Method (SORM)," Report, July 1986.

[8] Y.-T. Wu and O.H. Burnside, "Computation of Probability of Instability Using the Fast Probability Integration Method," Manuscript, July 1986.

[9] P. Thoft-Christensen and M.J. Baker, "Structural Reliability Theory and Applications," Springer-Verlag, 1982.

APPENDIX A

NESSUS PFEM Verification Studies

Dr. K.R. Rajagopal

Rocketdyne Division, Rockwell International

NESSUS PFEM VERIFICATION STUDIES
PROGRAM PLAN

- BASIC PHILOSOPHY
  - USE A WIDE RANGE OF CAPABILITIES THAT IS BEING DEVELOPED IN NESSUS
  - DUE TO BUDGETARY AND CODE DEVELOPMENT SCHEDULE CONSTRAINTS, ALL OPTIONS CANNOT BE EXERCISED AND VERIFIED FOR ALL COMPONENTS
  - SELECT THE AREA OF EMPHASIS, RANDOM VARIABLES AND ANALYSIS OPTIONS CONSISTENT WITH A REALISTIC HARDWARE ANALYSIS AND THE STATE OF CODE DEVELOPMENT

NESSUS PFEM VERIFICATION STUDIES
FIRST COMPONENT - TURBINE BLADE

- AREA OF EMPHASIS
  - STATIC ANALYSIS AND FREQUENCY EXTRACTION
- STATIC ANALYSIS
  - STRESS CALCULATION AT CHOSEN CRITICAL POINTS WHERE WE HAVE HAD A HISTORY OF CRACKING OF THE BLADE
  - IF POSSIBLE SLIP SYSTEM SHEAR STRESSES WILL BE EVALUATED
- DYNAMIC ANALYSIS
  - FREQUENCY

NESSUS PFEM VERIFICATION STUDIES
FIRST COMPONENT - TURBINE BLADE

- CHOSEN RANDOM VARIABLES
  - MATERIAL PROPERTY AND ORIENTATION
  - NODAL COORDINATES
  - TEMPERATURES, PRESSURE, CENTRIFUGAL LOAD

NESSUS PFEM VERIFICATION STUDIES
FIRST COMPONENT - TURBINE BLADE

- MODEL
  - A COMPARATIVELY LARGE SOLID MODEL
  - ABOUT 5000 DOF
  - IF IT CANNOT BE FIT IN THE COMPUTER CORE, OTHER OPTIONS FOR SMALLER MODELS MUST BE INVESTIGATED

NESSUS PFEM VERIFICATION STUDIES
SECOND COMPONENT - HIGH PRESSURE DUCT

- AREA OF EMPHASIS
  - STATIC AND FORCED RESPONSE ANALYSIS

- STATICS ANALYSIS
  - EVALUATION OF CRITICAL STRESSES
  - RANDOM VARIABLES (LOADS)
    - ACCELERATION LOADS
    - GIMBAL LOADS
    - MISALIGNMENT LOADS
    - FLOW LOADS

NESSUS PFEM VERIFICATION STUDIES
SECOND COMPONENT - HIGH PRESSURE DUCT

- DYNAMICS ANALYSIS
  - RANDOM VARIABLES
    - DAMPING
    - SUPPORT VIBRATION (PSD)
      - VARIATION OF RANDOM PART OF PSD
      - VARIATION OF SINUSOIDS WITH RESPECT TO AMPLITUDE AND FREQUENCY
    - DEGREE OF CORRELATION BETWEEN TWO SUPPORT VIBRATION LEVELS

NESSUS PFEM VERIFICATION STUDIES
THIRD COMPONENT - LOX POST

- AREA OF EMPHASIS
  - STATIC MATERIAL NONLINEAR ANALYSIS AND DYNAMICS
- STATICS
  - EVALUATION OF TOTAL PLASTIC STRAINS IN THE THREAD REGION OF LOX POST DUE TO THERMAL GRADIENT
  - RANDOM VARIABLES
    - NONLINEAR STRESS STRAIN LAW
    - TEMPERATURE
  - MODEL
    - 2-D AXISYMMETRIC

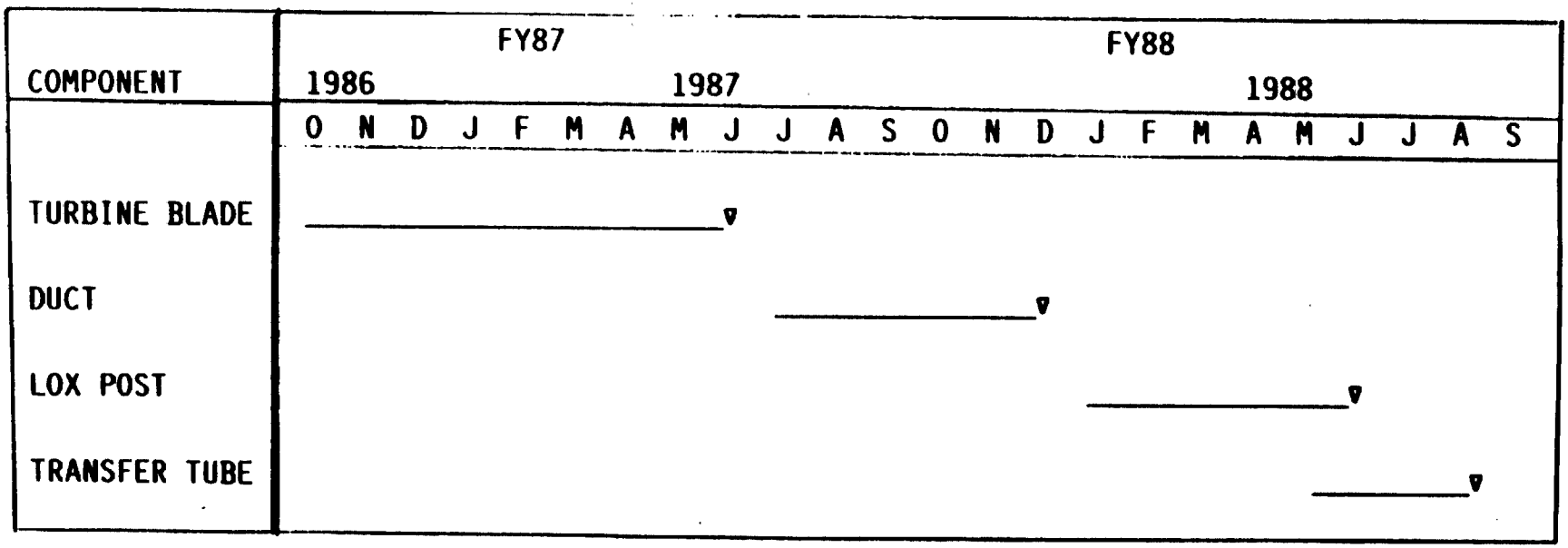
NESSUS PFEM VERIFICATION STUDIES
THIRD COMPONENT - LOX POST

- DYNAMICS
  - EVALUATION OF ALTERNATING STRESSES
  - RANDOM VARIABLES
    - PRESSURE
    - BOUNDARY CONDITIONS AT THREADS (HINGED TO FIXED)
  - MODEL
    - 3-D BEAM MODEL WITH SHELLS FOR SHIELD

NESSUS PFEM VERIFICATION STUDIES
FOURTH COMPONENT - TRANSFER TUBE STRUCTURAL LINER

- AREA OF EMPHASIS
  - EVALUATION OF BUCKLING LOADS
    - PERTURBED EIGEN VALUES (BIFURCATION)
    - NONLINEAR PREBUCKLING PATH, LARGE DEFLECTION BUCKLING LOAD
- STATICS
  - EVALUATION OF BUCKLING PRESSURES FOR THE STRUCTURAL LINER
  - RANDOM VARIABLES
    - MATERIAL PROPERTIES
    - THICKNESS
    - NODE COORDINATES
    - TEMPERATURE
    - BOUNDARY CONDITION FLEXIBILITY
- MODEL
  - 3-D THIN SHELL MODEL

NESSUS PFEM VERIFICATION EFFORTS
PFEM SCHEDULE



APPENDIX B

A Study of the Stability of the Elastostatic Perturbation Algorithm for Structural Problems with Transverse Shear Constraints

J.B. Dias and S. Nakazawa

MARC Analysis Research Corporation

Introduction

An algorithm for the efficient computation of the elastostatic response of a perturbed system discretized with finite elements has been proposed [2] and implemented in the NESSUS code as part of the PSAM development effort. Although this algorithm has been successfully used in sensitivity studies of several structural systems with random parameters, recent experience has indicated loss of stability for seemingly "small" perturbations in some problem classes. These problems typically involve approximate constraint equations which are embedded in the stiffness of the unperturbed problem, and perturbations which result in the modification of these constraint equations.

Finite element formulations for constrained problems using Lagrange multipliers and the penalty method have enjoyed widespread use in the recent past and have played an essential role in the development of successful methods for certain classes of problems. The literature on this subject is extensive and includes applications to:

- The analysis of plate and shell structures allowing shear deformation [7, 12, 16].
- Incompressible elasticity, e.g., rubber-like materials [5, 8].
- Deviatoric rate-independent plasticity [10].
- Incompressible flows, e.g., Stokes and Navier-Stokes equations [5, 14, 15], etc.

A fundamental assumption in the development of the perturbation algorithm in [2] is that the response of the unperturbed system constitutes a "good" approximation to the response of the perturbed system. This will not, in general, be the case, if the prescribed perturbation results in a noticeable change in the constraint equations present in the unperturbed system. Violation of this condition will often result in loss of stability and failure to converge. Thus, the presence of constraint equations in the finite element formulation may impose limits in the size of some perturbation parameters which are not

immediately obvious. Additional analysis must then be performed in order to determine exactly what constitutes a "small" perturbation.

Perturbation methods based on Taylor series expansions about the unperturbed solution have also been proposed by several researchers [3, 4, 11] and it is natural to ask how these pathologies manifest themselves in the solutions obtained by these methods. It can be shown that the displacement correction obtained in the first iteration is identical to the first-order term in the Taylor series expansion, and the one obtained in the second iteration is identical to the second-order term in the series. Thus, the rate of convergence of the iterative algorithm is closely related to the errors resulting from truncation of the Taylor series. One advantage of the iterative algorithm in [2] is that an error estimate (the force residual) must be computed and is available at every step of the iteration.

The Transverse Shear Constraint

The classical Poisson-Kirchhoff theory of plates requires C^1 continuity of displacement, as does the classical Bernoulli-Euler beam theory. However, the development of compatible C^1 interpolations in multi-dimensional cases is not straightforward, and considerable efforts and ingenuity were invested in the development of the first generation of finite element formulations for thin plate and shell problems [13].

In recent years, the Reissner-Mindlin theory of plates, which can accommodate transverse shear strains, has enjoyed widespread use. In this formulation, only C^0 continuity of displacements is required, allowing the construction of far simpler and less restrictive interpolation schemes. As a result, finite element formulations for medium-thick plate and shell problems have been developed, which retain accuracy even for thin plate and shell situations [7, 12, 16]. However, as the thin limit is approached, the "pure bending" modes dominate the solution, resulting in the emergence of penalty constraint terms in the stiffness equations.

The fundamental aspects of the problem may be observed in the one-dimensional analog of the Reissner-Mindlin plate theory, i.e., the Timoshenko beam. Thus, the Timoshenko beam theory may be used as a simpler, more manageable model exhibiting the pathologies that afflict Reissner-Mindlin plate theory. The total potential energy, including shear deformation, for an elastic beam of rectangular cross-section with thickness t and width b may be written as

$$\Pi = \frac{1}{2}\int_0^L \frac{Ebt^3}{12}\left(\frac{d\theta}{dx}\right)^2 dx + \frac{1}{2}\int_0^L \frac{\kappa Gbt}{2}\left(\frac{dw}{dx} - \theta\right)^2 dx - \int_0^L wqdx \qquad (1)$$

In this form, the first integral corresponds to the "pure bending" energy in the beam, whereas the second integral represents the shear deformation energy, and the third and last term accounts for the work done by the applied transverse loading. As the thickness t is reduced, the bending stiffness ($Ebt^3/12$) will decrease much faster than the shear stiffness ($\kappa Gbt/2$). In the limit, the shear stiffness term becomes a Lagrange multiplier enforcing the condition that

$$\frac{dw}{dx} = \theta$$

which is the assumption made a priori in Bernoulli-Euler beam theory that the rotation is the derivative of the transverse displacement.

The Discretized Problem

The finite element formulation for the Timoshenko beam problem using linear interpolations for both translational and rotational degrees-of-freedom produces an element stiffness matrix of the form

$$k^e = \frac{EI}{h}\begin{bmatrix} 0 & 0 & 0 & 0 \\ & 1 & 0 & -1 \\ & & 0 & 0 \\ \text{symm.} & & & 1 \end{bmatrix} + \frac{\kappa GA}{h}\begin{bmatrix} 1 & \frac{h}{2} & -1 & \frac{h}{2} \\ & \frac{h^2}{4} & -\frac{h}{2} & \frac{h^2}{4} \\ & & 1 & -\frac{h}{2} \\ \text{symm.} & & & \frac{h^2}{4} \end{bmatrix} \qquad (2)$$

where h is the element length, E and G are the elastic and shear moduli, I and A are the cross-sectional moment of inertia and area, and κ is a shape-dependent factor to account for non-uniform shear distribution in the cross-section. The particular case of a rectangular cross-section corresponds to $I = bt^3/12$ and $A = bt$, where t is the beam thickness and b its width. In order to simplify the algebra, it is convenient to combine the bending and shear stiffness terms to obtain

$$k^e = \kappa GA\begin{bmatrix} \frac{1}{h} & \frac{1}{2} & -\frac{1}{h} & \frac{1}{2} \\ & (\frac{h}{4} + \frac{\alpha}{h}) & -\frac{1}{2} & (\frac{h}{4} - \frac{\alpha}{h}) \\ & & \frac{1}{h} & -\frac{1}{2} \\ \text{symm.} & & & (\frac{h}{4} + \frac{\alpha}{h}) \end{bmatrix} \qquad (3)$$

where α is the ratio $EI/(\kappa GA)$ with dimension length squared. For a rectangular cross-section, assuming $\kappa = 1$ and incompressible material with $E = 3G$, this ratio becomes $\alpha = t^2/4$.

Stability Conditions

The iterative perturbation algorithm proposed in [2] can be summarized by the following recursion relations:

$$\bar{K}\, \bar{du}^{(n+1)} = \bar{f} - \bar{K}\, \bar{u}^{(n)} \qquad (4a)$$

$$\dot{u}^{(n+1)} = \dot{u}^{(n)} + d\dot{u}^{(n+1)} \qquad (4b)$$

where the symbol $\dot{}$ is used to denote the perturbed quantities. The consistency of the algorithm is provided for in the computation of the right-hand-side of (4a), since the process is equivalent to the minimization of the residual

$$\dot{r}^{(n)} = \dot{f} - \dot{K}\,\dot{u}^{(n)}$$

which will be attained if $\dot{u}^{(n)} = \dot{u}$, the exact value of the perturbed response. Stability is achieved if each displacement correction $d\dot{u}^{(n+1)}$ is smaller (in an appropriate norm) than the preceeding term, $d\dot{u}^{(n)}$. Both conditions must be satisfied for the iteration to converge to the exact solution.

The stability conditions can best be discussed in terms of an amplification matrix, which is derived next. Consider the form of equation (4a) in two consecutive iterations

$$K\,d\dot{u}^{(n+1)} = \dot{f} - \dot{K}\,\dot{u}^{(n)} \qquad \text{Iteration (n)}$$

$$K\,d\dot{u}^{(n)} = \dot{f} - \dot{K}\,\dot{u}^{(n-1)} \qquad \text{Iteration (n-1)}$$

and subtract the second from the first to obtain

$$K\,d\dot{u}^{(n+1)} - K\,d\dot{u}^{(n)} = -\dot{K}(\dot{u}^{(n)} - \dot{u}^{(n-1)})$$

$$K\,d\dot{u}^{(n+1)} = K\,d\dot{u}^{(n)} - \dot{K}\,d\dot{u}^{(n)} \qquad (5)$$

Premultiplication by K^{-1} on both sides yields

$$d\dot{u}^{(n+1)} = (I - K^{-1}\,\dot{K})\,d\dot{u}^{(n)}$$

where

$$\mathbf{A} = (\mathbf{I} - \mathbf{K}^{-1}\,\dot{\mathbf{K}}) \tag{6}$$

is the desired amplification matrix. The iteration will be stable if the $d\dot{u}^{(n)}$ decrease monotonically, i.e., if the spectral radius of every eigenvalue of **A** is less than unity.

In this form, the determination of the spectrum of the amplification matrix **A** would require a considerable amount of computation. In general, the eigenvectors of the perturbed stiffness matrix $\dot{\mathbf{K}}$ will be different from the eigenvectors of the unperturbed stiffness **K**. This will result in a nonsymmetric amplification matrix **A**. In addition, the size and structure of the amplification matrix in this form is entirely problem-dependent, so that it does not easily lend itself to analysis for the general case.

Stability Analysis

In order to circumvent some of the problems raised in the preceeding section, a Von Neumann stability analysis is performed on the difference pattern corresponding to the assembled system of equations at a typical internal node. Similar techniques have been used in studies of the stability of transient time integration schemes and nonlinear solution algorithms [1, 9].

The fundamental concept underlying these techniques is straightforward, even though the detailed derivations often require extensive algebraic manipulations. First, a set of stiffness equations corresponding to a typical node is extracted from the assembled stiffness equations. For a one-dimensional uniform mesh of two-noded beam elements, this will be a set of two equations, relating the shear and moment at node k to the translations and rotations at nodes k-1, k and k+1. Considering the linearly interpolated Timoshenko beam element in (3), these equations become

$$\begin{Bmatrix} f_k \\ m_k \end{Bmatrix} = \kappa GA \begin{bmatrix} -\frac{1}{h} & -\frac{1}{2} & \frac{2}{h} & 0 & -\frac{1}{h} & \frac{1}{2} \\ \frac{1}{2}\left(\frac{h}{4} - \frac{\alpha}{h}\right) & 0 & 2\left(\frac{h}{4} + \frac{\alpha}{h}\right) & -\frac{1}{2}\left(\frac{h}{4} - \frac{\alpha}{h}\right) \end{bmatrix} \begin{Bmatrix} u_{k-1} \\ \theta_{k-1} \\ u_k \\ \theta_k \\ u_{k+1} \\ \theta_{k+1} \end{Bmatrix} \qquad (7)$$

where f_k and m_k are the transverse shear force and moment at node k, and u_k and θ_k are the transverse displacement and the rotation at that node.

In order to capture the characteristics of the assembled system for an arbitrary displacement vector, a sinusoidal displacement pattern of the form

$$\begin{Bmatrix} u_k \\ \theta_k \end{Bmatrix} = \begin{Bmatrix} u \\ \theta \end{Bmatrix} e^{i\omega k} \qquad (8)$$

is imposed on the nodes of the one-dimensional mesh. Here, u and θ are <u>complex constants</u> representing the relative magnitude and phase of the displacements, and $\omega = 2\pi h/l$ where l is the (arbitrary) wavelength of the prescribed sinusoidal displacement pattern. Hence, the value $\omega = 0$ corresponds to the two rigid-body modes (one translation and one rotation), and the value $\omega = \pi$ will result in two displacement patterns which alternate signs between consecutive nodes. Any compatible displacement configuration of the discrete system may therefore be obtained by appropriate combination of a number of these "basic modes" with different ω between 0 and π.

Substituting (8) into (7) and using a few trigonometric identities, the following expression for the effective stiffness at an arbitrary ω may be derived:

$$\begin{Bmatrix} f_k \\ m_k \end{Bmatrix} = KGA \begin{bmatrix} \frac{2}{h}(1 - \cos\omega) & i \sin\omega \\ -i \sin\omega & \frac{h}{2}(1 + \cos\omega) + \frac{2\alpha}{h}(1 - \cos\omega) \end{bmatrix} \begin{Bmatrix} u \\ \theta \end{Bmatrix} \quad (9)$$

$$= k(\omega)\, u$$

This relation may be regarded as a "condensed" counterpart of the global stiffness equations, corresponding to a known displacement pattern (mode). Since no assumptions have been made on the value of ω, the equation above must hold for all values of ω that are compatible with the prescribed boundary conditions.

The techniques outlined in the preceeding paragraphs may be used to construct a "condensed" counterpart of the algorithmic relation in (5) corresponding to a given value of ω:

$$k(\omega)\, \dot{du}^{(n+1)} = k(\omega)\, \dot{du}^{(n)} - \dot{k}(\omega)\, \dot{du}^{(n)} \quad (10)$$

An amplification matrix relating consecutive displacement corrections for a given mode may be obtained by premultiplying (10) by $k^{-1}(\omega)$ to obtain

$$\dot{du}^{(n+1)} = a\, \dot{du}^{(n)}$$

where

$$a(\omega) = (I - k^{-1}(\omega)\, \dot{k}(\omega)) \quad (11)$$

is the desired amplification matrix. Stability conditions associated with particular classes of perturbations of the one-dimensional beam mesh problem may then be derived from the study of the eigenvalues of (11).

An interesting class of perturbation problems involves the (not necessarily uniform) elongation of the mesh. In the thin limit, the transverse shear constraint will impose the condition that

$$\frac{dw}{dx} = \theta$$

on the displacement solution of the unperturbed system. However, in the perturbed (elongated) beam, a <u>different</u> transverse shear constraint is in effect, which is <u>not</u> satisfied by the displacement solution for the unperturbed system, i.e.,

$$\frac{dw}{dx} \neq \theta$$

From the form of (1), it is clear that a very large amount of shear deformation energy is generated when the displacement solution for the unperturbed problem is imposed on the perturbed system, even for seemingly "small" elongations of the mesh.

In order to obtain a stability limit for this class of problems, a uniform elongation of the mesh is considered. The element length on the perturbed mesh thus becomes $\bar{h} = h(1+\varepsilon)$, so that each element in the mesh is elongated by the same amount. It follows that $\mathbf{k}^{-1}(\omega)$ may be expressed as

$$\mathbf{k}^{-1}(\omega) = \frac{h^2}{4\alpha} \frac{1}{(1 - \cos\omega)^2} \begin{bmatrix} \frac{h}{2}(1+\cos\omega) + \frac{2\alpha}{h}(1-\cos\omega) & -i\sin\omega \\ i\sin\omega & \frac{2}{h}(1-\cos\omega) \end{bmatrix} \quad (12)$$

and $\bar{\mathbf{k}}(\omega)$ as

$$\bar{\mathbf{k}}(\omega) = \begin{bmatrix} \frac{2}{h(1+\varepsilon)}(1-\cos\omega) & i\sin\omega \\ -i\sin\omega & \frac{h(1+\varepsilon)}{2}(1+\cos\omega) + \frac{2\alpha}{h(1+\varepsilon)}(1-\cos\omega) \end{bmatrix} \quad (13)$$

The resulting amplification matrix will be

$$a(\omega) = \begin{bmatrix} \frac{\varepsilon}{1+\varepsilon} + \frac{\varepsilon}{1+\varepsilon}\left(\frac{h^2}{4\alpha}\right)\cot^2\left(\frac{\omega}{2}\right) & i\varepsilon\left(\frac{h^2}{8\alpha}\right)\cot^3\left(\frac{\omega}{2}\right) - i\frac{\varepsilon}{1+\varepsilon}\left(\frac{h}{2}\right)\cot\left(\frac{\omega}{2}\right) \\ i\frac{\varepsilon}{1+\varepsilon}\left(\frac{h}{2\alpha}\right)\cot\left(\frac{\omega}{2}\right) & \frac{\varepsilon}{1+\varepsilon} - \varepsilon\left(\frac{h^2}{4\alpha}\right)\cot^2\left(\frac{\omega}{2}\right) \end{bmatrix} \quad (14)$$

and has eigenvalues of the form

$$\lambda_1(\omega) = \frac{\varepsilon}{1+\varepsilon} - \frac{1}{2}\frac{\varepsilon^2 f^2}{1+\varepsilon}\left(1 - \sqrt{1 + \frac{4}{\varepsilon^2 f^2}}\right) \quad (15a)$$

$$\lambda_2(\omega) = \frac{\varepsilon}{1+\varepsilon} - \frac{1}{2}\frac{\varepsilon^2 f^2}{1+\varepsilon}\left(1 + \sqrt{1 + \frac{4}{\varepsilon^2 f^2}}\right) \quad (15b)$$

where

$$f^2 = \left(\frac{h^2}{4\alpha}\right)\cot^2\left(\frac{\omega}{2}\right) \quad (15c)$$

All values of ω which are relevant to the analysis of the discrete system lie between 0 and π. The highest deformation modes representable by the discretized system correspond to $\omega = \pi$, which will result in

$$\lambda_1 = \lambda_2 = \frac{\varepsilon}{1+\varepsilon}$$

Thus, even for relatively large ε, the spectral radius of the amplification matrix will be less than unity and the corresponding deformation modes remain stable. At the opposite end of the spectrum lie the rigid body modes, corresponding to $\omega = 0$. As ω approaches 0, the value of $\cot\left(\frac{\omega}{2}\right)$ becomes unbounded. This means that the rigid body modes are unconditionally unstable for any nonzero value of ε, as expected.

An asymptotic analysis of the eigenvalues of the amplification matrix for large values of ε will yield

$$\lambda_1 \approx 1$$

$$\lambda_2 \approx -\frac{\varepsilon^2 f^2}{1+\varepsilon^2}$$

From the behavior of $\cot(\frac{\omega}{2})$, it can be concluded that f may become quite large for small values of ω. This will result in $|\lambda_2| > 1$ even for seemingly small ε, and will cause the associated deformation mode to grow unbounded. It should be emphasized, however, that the asymptotic expressions above typically represent reasonable approximations to the eigenvalues of $\mathbf{a}(\omega)$ only for values of ε well above the stability limit and cannot be used to approximate the eigenvalues within the stability bounds.

From the above discussion, it is clear that the higher deformation modes (with small values of ω) will govern the stability of the algorithm. This is in contrast with the well known results for the stability of explicit time integration algorithms in dynamics, which are governed by the highest frequency modes present in the discretized system. Any attempts at enhancing the stability of the perturbation algorithm must therefore take into account the fact that the displacement modes which require stabilization are among the most needed to represent the response of the perturbed system. Stability in the higher deformation modes must not compromise the accuracy of these modes, which rules out the use of conventional stabilization procedures.

A Numerical Example

A test problem was set up using a one-dimensional mesh of ten Timoshenko beam elements (NESSUS Element 98) with h = 2.00 and t = 0.25, and made of incompressible material (ν = 0.50). Three different cases were analyzed, corresponding to the following boundary conditions:

1. Cantilever beam. In this case, the lovest displacement mode has a wavelength of four times the beam length, corresponding to $\omega=\pi/20$.
2. Beam with both ends pinned. The wavelength of the lovest mode is twice the beam length, corresponding to $\omega=\pi/10$.
3. Beam with both ends fixed. The wavelength of the lovest mode is equal to the beam length, corresponding to $\omega=\pi/5$.

The variation of the spectral radius of the eigenvalues of the amplification matrix in the lowest mode as a function of the perturbation parameter ε is shown in Figures 3 to 5. In all cases, loss of stability was observed at the value of ε corresponding to a spectral radius of 1.00, as predicted by the analysis. Similar behavior has been observed using a shell model (NESSUS Element 75) of the same problem.

Conclusions

The stability conditions for the elastostatic perturbation algorithm proposed in [2] have been described. A Von Neumann stability analysis of the algorithm was performed for the case of a uniform one-dimensional mesh of linearly interpolated Timoshenko beam elements. Closed form expressions for the stability limit in terms of the perturbation parameter ε were derived for the case of uniform elongation of the mesh by a factor of $(1+\varepsilon)$. This form of perturbation has been observed to result in loss of stability with the current implementation of the algorithm even for seemingly small values of the perturbation parameter ε. The stability limits predicted by the analysis are in full agreement with those observed in numerical experiments on one-dimensional meshes of linearly interpolated beam and plate elements.

The development of general closed-form results for unstructured, multi-dimensional meshes subjected to non-uniform distortion does not appear to be practical. However, limited experience has indicated that the results for the uniform mesh can be used to obtain a conservative estimate to the stability limit for a more general mesh. Therefore, the development of "smart" algorithms to adaptively adjust the size of the

perturbation parameter in order to ensure convergence appears very promising.

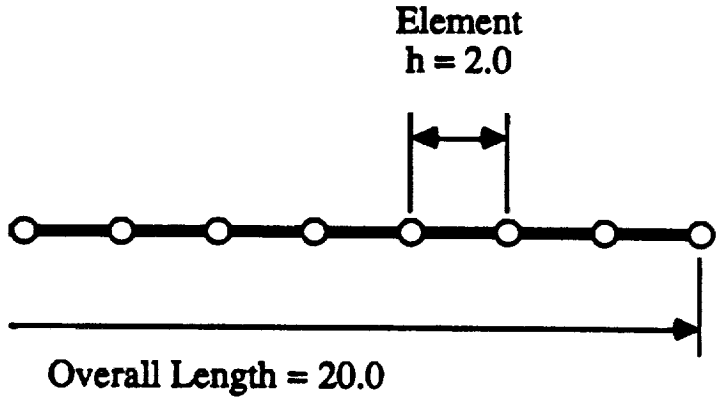


FIGURE 1: Mesh for the One-Dimensional Model Problem

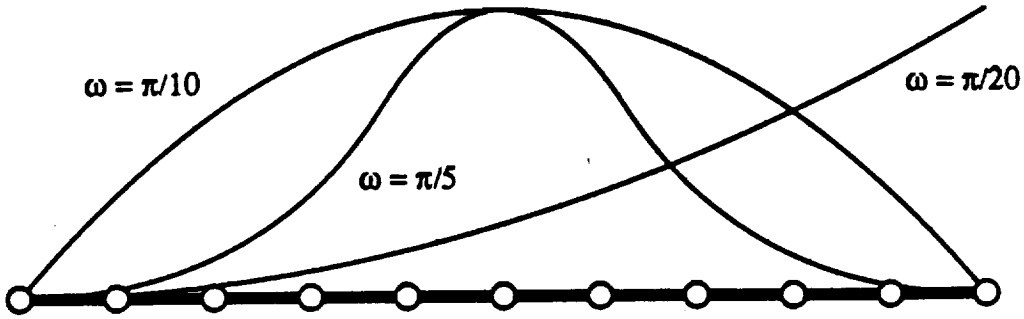


FIGURE 2: Typical Displacement Patterns for the Values of ω Used in the Numerical Example

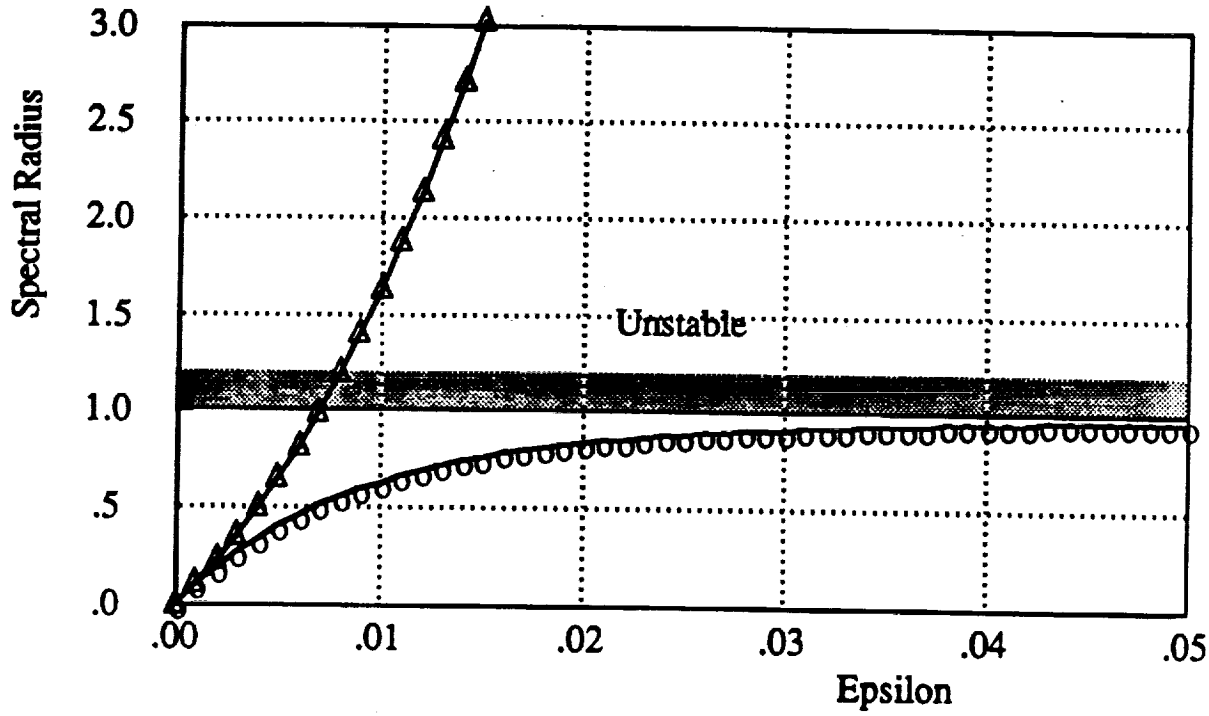


FIGURE 3: Spectral Radius of the Amplification Matrix as a Function of the Perturbation Parameter ε for $h/t = 8.00$ and $\omega = \pi/20$

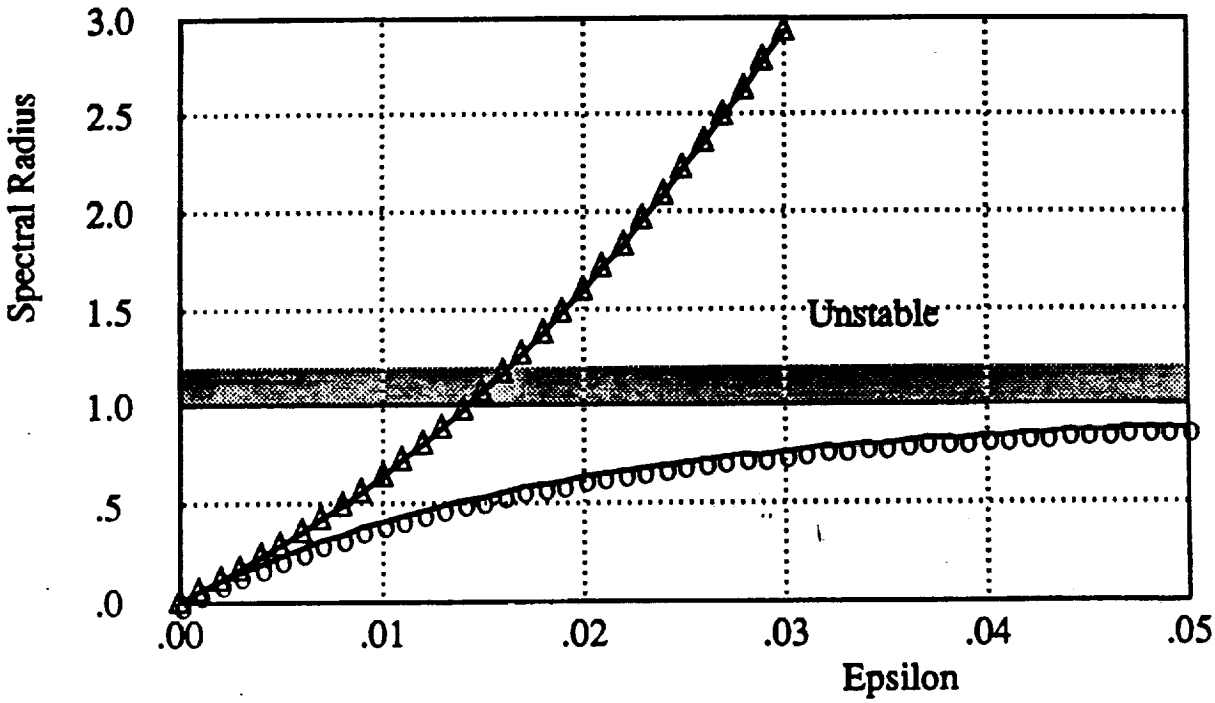


FIGURE 4: Spectral Radius of the Amplification Matrix as a Function of the Perturbation Parameter ε for $h/t = 8.00$ and $\omega = \pi/10$

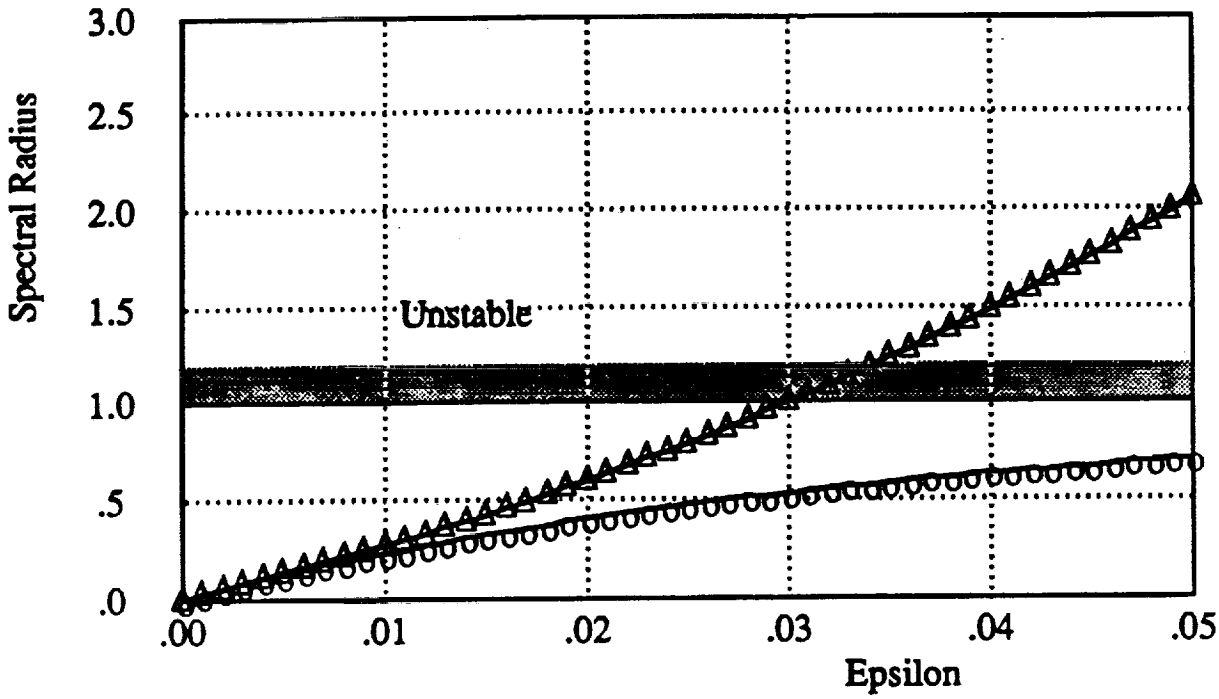


FIGURE 5: Spectral Radius of the Amplification Matrix as a Function of the Perturbation Parameter ε for $h/t = 8.00$ and $\omega = \pi/5$

APPENDIX C

Improvements to Approximate Forms of Certain Functions In the Wu/FPI Code

Sueng J. Lee
Jack T.L. Chang
Paul H. Wirsching

The University of Arizona

1.0 DISCUSSION

In private correspondence of January 13, 1986 with Y. N. Chen of the American Bureau of Shipping, we were informed that certain numerical algorithms in the Wu/FPI code lacked the precision to ensure "small" errors in resulting point probability estimates. Those subroutines for which improvements were suggested were:

1. The normal CDF
2. The inverse normal CDF
3. The gamma function
4. The shape parameter of the gamma function
5. The EVD parameters

ABS implimented improved numerical procedures in FPI and studied several examples. Because the differences in point probability estimates observed by ABS in old FPI and their new version seemed significant, a study was undertaken to carefully examine the approximate forms and to introduce improvements where appropriate. The improvement in the Euler constant (for EVD parameters) for 8 digit accuracy was trivial and was implimented immediately. Numerical algorithms for the other terms cited above were developed. Their performance was carefully examined. A detailed description of the approximation forms and their behavior is presented in Chapter 2.

The forms presented in Chapter 2 were introduced into FPI, replacing their less accurate counterparts. FPI analysis using the old and new code was performed on several examples. The results are summarized in Chapter 3.

Differences in the results of the old and new code are far less than observed by ABS in their version of the code. At this time, there is no explanation for the discrepancy

2.0 APPROXIMATE FORMS OF FUNCTIONS USED IN PROBABILITY CALCULATIONS

2.1 Gamma Function

Ref: Abramowitz, Handbook of Mathematical Functions, NBS.

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t}\, dt, \quad x > 0$$

The Asymptotic Formula

$$\ln \Gamma(x) \tilde{=} (x - \tfrac{1}{2}) \ln x - x + \tfrac{1}{2} \ln (2\pi) + \frac{1}{12x} - \frac{1}{300x^3}$$

$$+ \frac{1}{1260x^5} - \frac{1}{1680x^7} +$$

$$(x \to \infty \text{ in } |\text{Arg } x| < \pi)$$

After testing this formula, we found that when $x \geq 6$, ten digit accuracy is provided. If x is increased, this form is even more accurate.

In this program, x is divided into two parts $X \geq 6$, and $0 < x < 6$. If $x \geq 6$, use the asymptotic formula directly. If $0 <$ < 6, then let

$$N = \text{INTEGER } (x)$$

$$Z = 6 - N + x$$

and calculate $\ln \Gamma(Z)$ using the asymptotic formula.

Then let, $\ln \Gamma(x) = \ln \Gamma(Z) - \sum_{J=1}^{6-N} \ln(x + J - 1.0)$

Example $x = 1.9$

$$Z = 6. - 1 + 1.9 = 6.9$$

$$\ln \Gamma(1.9) = \ln \Gamma(6.9) - (\ln (1.9) + \ln (2.9) + \ln (3.9) + \ln (4.9) + \ln (5.9)$$

$$= \ln \frac{\Gamma(6.9)}{(1.0)(2.9)(3.9)(4.9)(5.9)}$$

$$\Gamma (1.9) = \frac{\Gamma(6.9)}{(1.9)(2.9)(3.9)(4.9)(5.9)}$$

If more accuracy is needed, then increase the 6 in the above algorithm to 7, 8, or a larger number

Fig. 1 Flowchart for Gamma Function Approximation

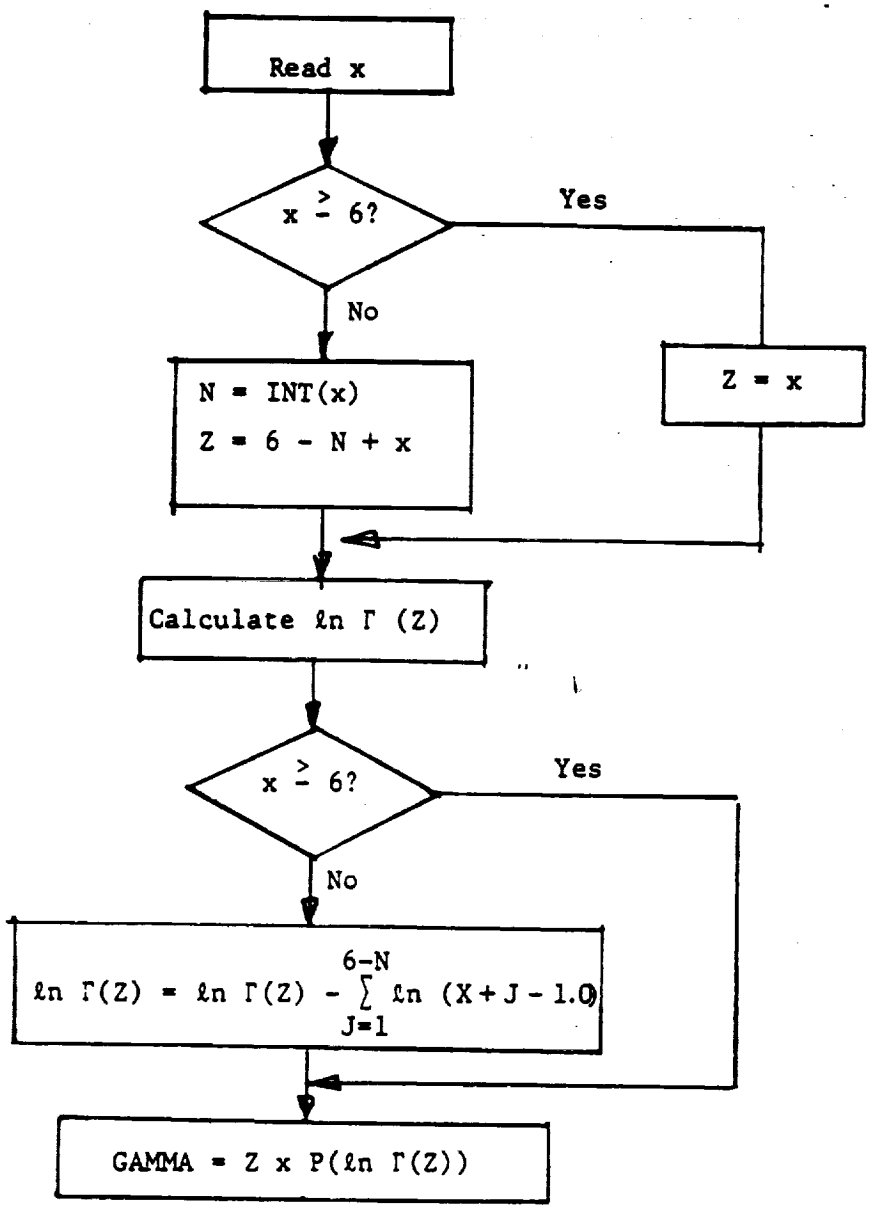


Table 1. Performance of Gamma Function Approximation

x	Asymptotic formula	Exact
1.0	.9999999999	1.0
1.1	.9513507698	.9513507699
1.2	.9181687423	.9181687424
1.3	.8974706962	.8974706964
1.4	.8872638174	.8872638175
1.5	.8862269254	.8862269255
1.6	.8935153492	.8935153493
1.7	.9086387328	.9086387329
1.8	.9313837709	.9313837710
1.9	.9617658318	.9617658311
2.0	.9999999999	1.0

Subroutine for Gamma Function

```
      DOUBLE PRECISION FUNCTION GAMMA(Y1,PI)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      X=Y1+1.D+0
      Z=X
      IF(X.GE.6.0D+0)GO TO 456
      N=INT(X)
      Z=(6.0D+0)-N+X
456   Y=1.D+0/Z**2
      ALG=(Z-.5D+0)*DLOG(Z)+.5D+0*DLOG(PI*2.D+0)-
     $ Z-(1.D+0/(12.D+0*Z))*(((Y/0.14D+3-1.D+0/0.105D+3)*Y+
     $ 1.D+0/.3D+2)*Y-1.D+0)
      IF(X.GE.6.D+0)GO TO 457
      ITE=6-N
      DO 3 J=1,ITE
      A=X+J-1.D+0
      ALG=ALG-DLOG(A)
3     CONTINUE
457   GAMMA=DEXP(ALG)
      RETURN
      END
```

2.2 Bisection Method for the Shape Parameter α of the Weibull Distribution

The coefficient of variation C_X in terms of the shape parameter α of the Weibull distribution is given as

$$C_X = \sqrt{\frac{\Gamma^2(1+\alpha)}{\Gamma(1+2\alpha)} - 1}$$

Given C_X, it is required to compute α.

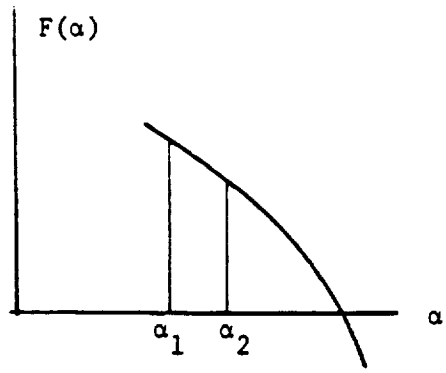
Define

$$F(\alpha) = -(1 + C_X^2)\,\Gamma^2(1+\alpha) + \Gamma(1+2\alpha)$$

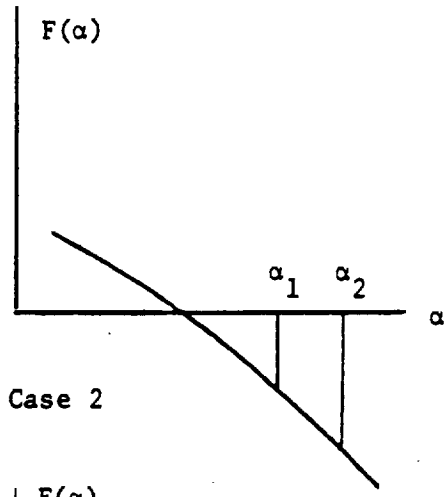
Approximate $\alpha_1 = (C_X)^{1.08}$. Then calculate $F(\alpha_1)$, and let $\alpha_2 = \alpha_1 + .1$. Calculate $F(\alpha_2)$ and let $F12 = F(\alpha_1) * F(\alpha_2)$.

If $F12 \leq 0$; we know that the root will be bracketed by α and α_2. Then use the general bisection method as described below.

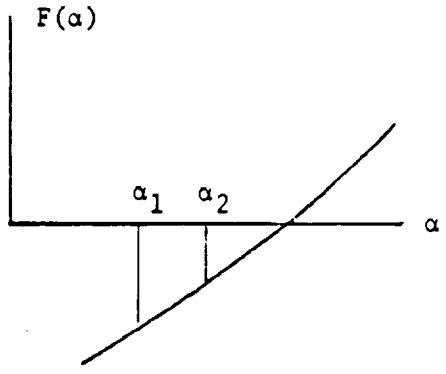
If $F12 > 0$, there are four possible cases.



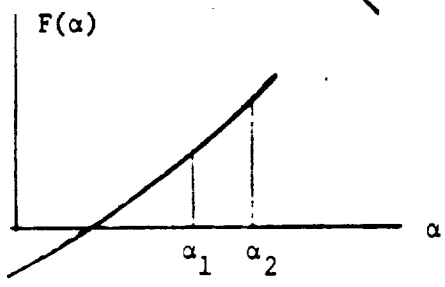
Case 1



Case 2



Case 3



Case 4

If the function looks like Case 1 or Case 3, then let $\alpha_1 = \alpha_2$; $\alpha_2 = \alpha_2 + 0.$

If the function looks like Case 2 or Case 4, then let $\alpha_2 = \alpha_1$; $\alpha_1 = \alpha_1 - 0.$

Then calculate F12 until F12 < 0, at which time the bisection method can be used

(A) General Bisection Method

1. $\alpha_3 = \dfrac{\alpha_1 + \alpha_2}{2}$ (*)

$F13 = F(\alpha_1) * F(\alpha_3)$

If F13 < 0, $\alpha_2 = \alpha_3$

If F13 > 0, $\alpha_1 = \alpha_3$

If $|\alpha_1 - \alpha_2| > 10^{-7}$ go to (*) and repeat.

$\leq 10^{-7}$ STOP; Let $\alpha = \alpha_1$

(B) Performance

Consider the Rayleigh Distribution

$C_X = .522723201$

	Asymptotic formula	Exact
α	2.00000014531220	2.0

Program: Bisection Method for Weibull Shape Parameter

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      F(X,COV,PI)=-(1.D0+COV**2)*GAMMA(X,PI)**2+GAMMA(2.*X,PI)
      PI=4.D0*DATAN(1.D0)
      COV=.522723201
      X1=COV**(1.09)
7     F1=F(X1,COV,PI)
      IF(DABS(F1).LE.1.D-7) GO TO 1
      X2=X1+.1D0
      F2=F(X2,COV,PI)
      F12=F1*F2
      IF(F12.LT.0.) GO TO 20
      IF(F1.GT.0..AND.F2.GT.F1) X1=X1-.1D0
      IF(F1.LT.0..AND.F2.GT.F1) X1=X2
      IF(F1.GT.0..AND.F1.GT.F2) X1=X2
      IF(F1.LT.0..AND.F1.GT.F2) X1=X1-.1D0
      GO TO 7
20    CONTINUE
2     X3=(X1+X2)*.5D0
      F13=F(X1,COV,PI)*F(X3,COV,PI)
      IF(F13.LT.0.) X2=X3
      IF(F13.GT.0.) X1=X3
      DX=DABS(X1-X2)
      IF(DX.GE.1.D-7) GO TO 2
1     ALPHA=1.D0/X1
      WRITE(*,*) ' ALPHA = ',ALPHA
      ZZ=.95D0
      DO 1000 I=1,21
      ZZ=ZZ+.05D0
      WRITE(*,*) ZZ,GAMMA(ZZ-1.D0,PI)
1000  CONTINUE
      STOP
      END
```

2.3 CDF of Normal Distribution

Ref: Abramowitz: Handbook of Mathematical Functions, NBS

$$P = \Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2} dt$$

$$\simeq \begin{cases} 1 - Z(x)(b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5) & \text{if } x \geq 0 \\ Z(x)(b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5) & \text{if } x < 0 \end{cases}$$

Where,

$$Z(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

$$t = \frac{1}{1 + px}$$

$$p = 0.231621$$

$$b_1 = 0.319381530$$

$$b_2 = -.356563782$$

$$b_3 = 1.781477947$$

$$b_4 = -1.821255978$$

$$b_5 = 1.330274420$$

This approximation is advertised to produce errors in P of less than 10^{-7}. (See performance on Table 2, p. 15.) When $x > 0$, it appears that this level of accuracy is being realized. For the very small P values associated with $x < 0$, errors are somewhat larger. But the operational range for structural reliabilit analysis is $-5 < x < 5$, and at worst we are getting four place accuracy.

It is important to note that we cannot verify the accuracy of column (4) in Table 2, p. 15. During this investigation, some anomolies were discovered in the Abramowitz table.

Program: Standard Normal CDF

```
      PROGRAM CDFPDF
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /TWO/ PI,SPI2
      PI=4.D0*DATAN(1.D0)
      SPI2=1./(DSQRT(2.D0*PI))
      X=-11.D0
      DO 1 I=1,22
      X=X+1.D0
      PHI=CDFNOR(X)
      XPHI=XINV(PHI)
      WRITE(*,*) X,PHI,XPHI
1     CONTINUE
      STOP
      END
      DOUBLE PRECISION FUNCTION CDFNOR(Z)
C THIS FUNCTION COMPUTES THE NORMAL CDF.
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /TWO/ PI,SPI2
     DATA A/0.31938153D0/,B/-0.356563782D0/,C/1.781477937D0/,
    +D/-1.821255978D0/,E/1.330274429D0/
     EZ=-(Z**2)*.5D0
     CDFNOR=0.0D0
     IF(EZ.LE.-200.0D0) GO TO 1
     ZX=SPI2*DEXP(EZ)
       IF(DABS(Z).GT.6.D0) GO TO 2
     T=1.D0/(1.D0+(0.2316419D0*DABS(Z)))
     CDFNOR=ZX*T*(A+T*(B+T*(C+T*(D+T*E))))
       GO TO 1
2     Z2=1.D0/(Z*Z)
      CDFNOR=ZX*(1.D0-Z2*(1.D0-3.D0*Z2*(1.D0-5.D0*Z2)))/DABS(Z)
  1  IF(Z.GT.0.0D0) CDFNOR=1.0D0-CDFNOR
     RETURN
     END
```

2.4 Bisection Method for the Inverse Normal CDF, $x = \Phi^{-1}(P)$

Ref: Abramowitz, Handbook of Mathematical Functions, NBS

First, the following method is used to obtain an approximation to x.

$$x_1 = \Phi^{-1}(P) = t - \frac{C_o + C_1 t + C_2 t^2}{1 + d_1 t + d_2 t^2 + d_3 t^3}$$

where,

$$t = \sqrt{-2 \ln P}, \qquad 0 < P \leq .5$$

$$C_o = 2.515517$$

$$C_1 = .802853$$

$$C_2 = .010328$$

$$d_1 = 1.432788$$

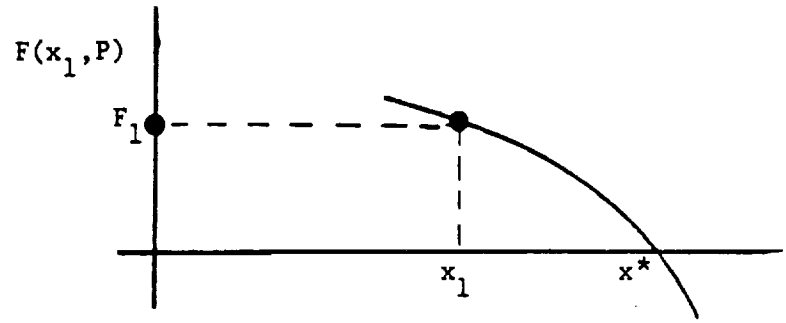
$$d_2 = .18926$$

$$d_3 = .001308$$

This approximation gives only four digit accuracy.

Define $F(x, P) = P - \Phi(x)$.

1. Let $x_1 = \Phi^{-1}(P)$ using the "crude" approximation above. Then F(x, P) looks like



2. Let $F_1 = F(x_1, P)$* (A)

If $F_1 > 0$, $x_2 = x_1 + .001$

If $F_1 < 0$, $x_2 = x_1 - .001$

If $F_1 = 0$, STOP

Then in the second iteration, let

$F_2 = F(x_2, P)$

Calculate $F12 = F(x_1, P) * F(x_2, P)$

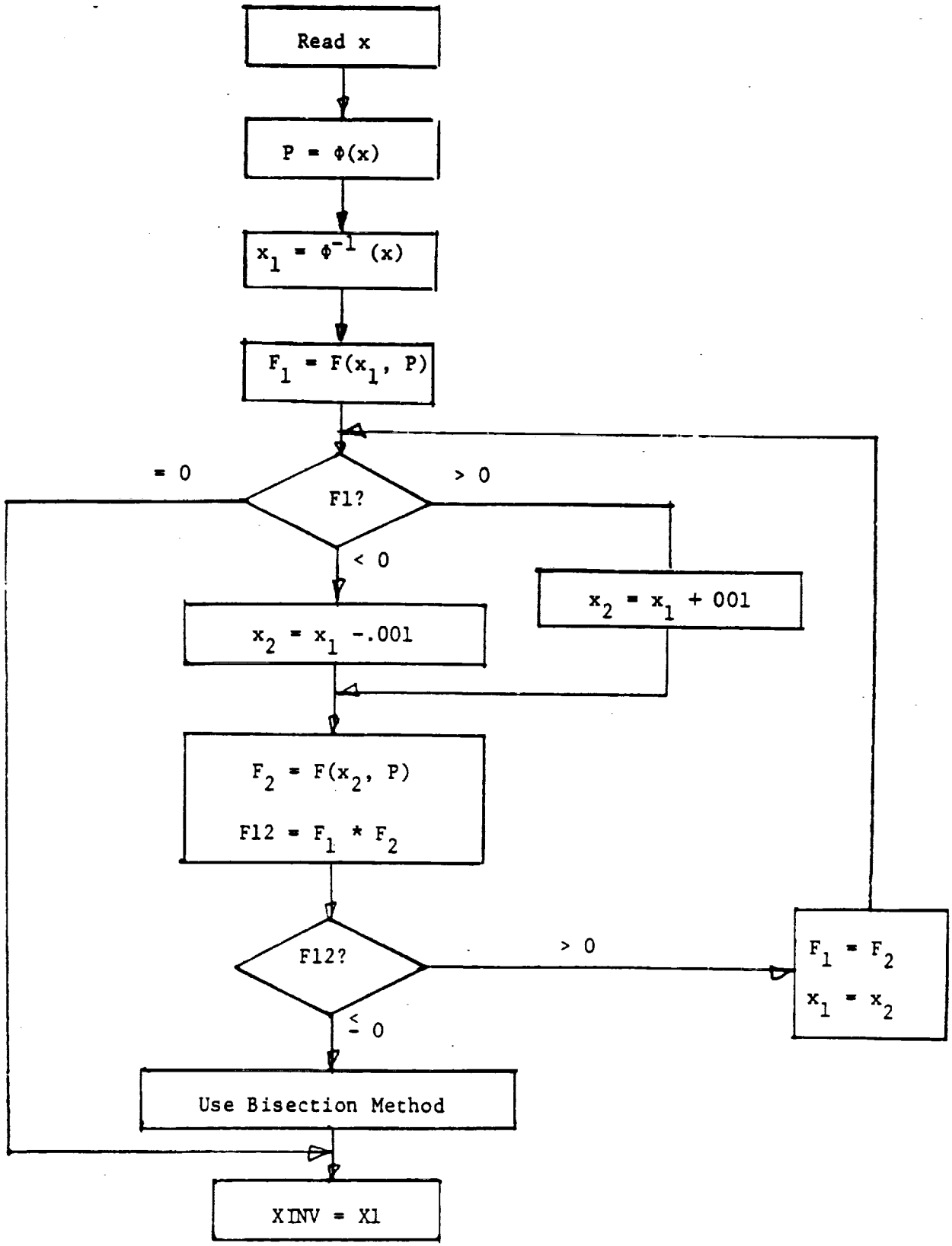
If $F12 < 0$, use general Bisection Method

If $F12 > 0$, then $X_1 = x_2$

$F_1 = F_2$

go back to (A) and repeat.

*The function $\Phi(\cdot)$ is obtained using the form of Sec. 2.3.



where $F(x, P) = P - \Phi(x)$

Fig. 2 Flow Chart for Inverse Normal Approximation

Table 2. Performance of Normal and Inverse cdf Approximations

(1) x	(2) P = $\Phi(x)$ using form for CDF (Abramowitz)	(3) $\Phi^{-1}(P)$ using bisection method	(4) P=$\Phi(x)$ exact as published in Abramowitz
-10	1.619845601 E-24	-9.99999999	7.6199 E-24
-8	6.220925810 E-16	-7.99999999	6.2210 E-16
-6	9.901218571 E-10	-5.99999999	9.8659 E-10
-4	3.168603459 E-5	-3.99999999	3.1671 E-5
-3	1.349967223 E-3	-3.00000000	1.349898032 E-3
-2	2.275006201 E-2	-2.00000000	2.275013195 E-2
-1	1.586552595 E-1	-.99999999	1.586552540 E-1
0	4.999999994 E-1	3.61190816 E-11	5.0000 E-1
1	8.413447404 E-1	.99999999	8.413447460 E-1
2	9.772499379 E-1	2.00000000	9.772498680 E-1
3	9.986500327 E-1	3.00000000	9.986501019 E-1
4	9.999683139 E-1	3.99999999	9.999683288 E-1
6	9.999999990 E-1	4.99999998	
8	1.0	7.9911351772922	

Approximate form as described in Sec. 2.3

The inverse is obtained using the column (2) values with the algorithm described in Sec. 2.4. Columns (1) and (3) should compare.

Column (4) and column (1) should compare. See comments on p. 10.

Program: Bisection Method for Inverse Standard Normal CDF

```
      DOUBLE PRECISION FUNCTION XINV (Z)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        F(X,P1)=P1-CDFNOR(X)
      Y=Z
      IF(Z.GT.0.5D0) Y=1.D0-Z
        IF(Z.EQ.1.D0) STOP
      C0=2.515517D0
      C1=0.802853D0
      C2=0.010328D0
      D1=1.432788D0
      D2=0.189269D0
      D3=0.001308D0
        T=(-2.D0*DLOG(Y))**.5D0
      DNUM=C0+T*(C1+T*C2)
      DNOM=1.0D0+T*(D1+T*(D2+T*D3))
      X=T-(DNUM/DNOM)
      IF(Z.LT.0.5D0) X=-X
        X1=X
        F1=F(X,Z)
   80   IF(F1.GT.0.D0) X2=X1+.001D0
        IF(F1.LT.0.D0) X2=X1-.001D0
        IF(F1.EQ.0.D0) GO TO 2
        F2=F(X2,Z)
        F12=F1*F2
        IF(F12.LE.0.D0) GO TO 8
        X1=X2
        F1=F2
        GO TO 80
    8   X3=(X1+X2)*.5D0
        F13=F(X1,Z)*F(X3,Z)
        IF(F13.LE.0.D0) X2=X3
        IF(F13.GT.0.D0) X1=X3
        DX=DABS(X1-X2)
        IF(DX.GT.1.D-10) GO TO 8
    2   XINV=X1
        RETURN
        END
```

3.0 EXAMPLES COMPARING OLD AND NEW FPI

Following are several examples for which comparisons of results from old and new FPI are presented. These examples were those studied in an AME master's report by Jack T. L. Chang entitled, "Investigation of the Wu Algorithm for Computing Structural Reliability" (October 1985). In summary, introduction of the improved algorithms did not significantly alter the results, at least for the examples considered.

Results for the improved ABS FPI program for those examples considered are given in parentheses. In Examples 4 and 5, the results of the improved versions of the ABS and the UA codes differ significantly. There is at this time no explanation for the disagreement. An efficient Monte Carlo code for point probability estimates is under development. It will be able to check FPI calculations, but because the same numerical algorithms will be in both UA codes, the comparisons may not resolve this issue.

EXAMPLE 1 Note: ABS program results in parentheses

-DATA

FAILURE FUNCTION : $g = R - (L + D)$

FAILURE EVENT : $g < 0$

VARIABLE	DISTRIBUTION	MEAN/MEDIAN*	STD. DEV.	C.O.V.
R	WEIBULL	50.	5.0	0.1
L	EVD	10.	2.0	0.2
D	LOGNORMAL	20. *	3.034	0.15

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	2.768	2.783 (2.783)	0.54
	Pf	2.821 E-3	2.6931 E-3 (2.6931 E-3	4.79
Wu/FPI	β	2.692	2.707 (2.680)	0.55
	Pf	3.554 E-3	3.398 E-3 (3.682 E-3)	4.59
Monte Carlo(4)	Pf	3.600 E-3		

(1) Developed by C. Kelly, Y. T. Wu
(2) With improvements to numerical algorithms: (a) gamma function, (b) Weibull shape parameter, (c) standard normal cdf, (d) inverse normal cdf, (e) EVD parameters
(3) Assumes new program is exact
(4) Does not have improvements in numerical algorithms

EXAMPLE 2 Note: ABS program results in parentheses

-DATA

FAILURE Function : $g = \frac{\sigma_f'}{E} (2N)^b + \epsilon_f' (2N)^c - \epsilon_s$

FAILURE EVENT : $g < 0$

VARIABLE	DISTRIBUTION	MEAN/MEDIAN*	STD. DEV.	C.O.V.
ϵ_s	EVD	0.0015	0.00015	0.1
σ_f'	LOGNORMAL	310.0 *	145.10	0.43
ϵ_f'	LOGNORMAL	9.14 *	0.458	0.05

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	2.881	2.881 (2.881)	0.00
	Pf	1.981 E-3	1.983 E-3 (1.983 E-3)	0.10
Wu/FPI	β	2.851	2.850 (2.846)	0.04
	Pf	2.183 E-3	2.183 E-3 (2.215 E-3)	0.00
Monte Carlo(4)	Pf	2.123 E-3		

(1) Developed by C. Kelly, Y. T. Wu
(2) With improvements to numerical algorithms: (a) gamma function, (b) Weibull shape parameter, (c) standard normal cdf, (d) inverse normal cdf, (e) EVD parameters
(3) Assumes new program is exact
(4) Does <u>not</u> have improvements in numerical algorithms

EXAMPLE 3 Note: ABS program results in parentheses

-DATA

FAILURE FUNCTION : $g = X_1 + 2X_2 + 2X_3 + X_4 - 5(X_5 + X_6)$

FAILURE EVENT : $g < 0$

VARIABLE	DISTRIBUTION	MEAN/MEDIAN*	STD. DEV.	C.O.V.
X_1	LOGNORMAL	119.4 *	12.	0.1
X_2	LOGNORMAL	119.4 *	12.	0.1
X_3	LOGNORMAL	119.4 *	12.	0.1
X_4	LOGNORMAL	119.4 *	12.	0.1
X_5	LOGNORMAL	38.31 *	12.	0.3
X_6	LOGNORMAL	47.89 *	15.	0.3

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	2.348	2.348 (2.348)	0.00
	Pf	0.942 E-2	0.943 E-2 (0.943 E-2)	0.1
Wu/FPI	β	2.235	2.234 (2.256)	0.04
	Pf	1.274 E-2	1.274 E-2 (1.204 E-2)	0.00
Monte Carlo(4)	Pf	1.221 E-2		

(1) Developed by C. Kelly, Y. T. Wu
(2) With improvements to numerical algorithms: (a) gamma function, (b) Weibull shape parameter, (c) standard normal cdf, (d) inverse normal cdf, (e) EVD parameters
(3) Assumes new program is exact
(4) Does not have improvements in numerical algorithms

EXAMPLE 4 Note: ABS program results in parentheses

-DATA

FAILURE FUNCTION : $g = X_1 + X_2 + X_3 + X_4 + X_5 - Y_1 - Y_2 - Y_3 - Y_4 - Y_5$

FAILURE EVENT : $g < 0$

VARIABLE	DISTRIBUTION	MEAN/MEDIAN*	STD. DEV.	C.O.V.
X_1	WEIBULL	10.0	3.5	0.35
X_2	WEIBULL	10.0	3.5	0.35
X_3	WEIBULL	10.0	3.5	0.35
X_4	WEIBULL	10.0	3.5	0.35
X_5	WEIBULL	10.0	3.5	0.35
Y_1	EVD	5.0	1.75	0.35
Y_2	EVD	5.0	1.75	0.35
Y_3	EVD	5.0	1.75	0.35
Y_4	EVD	5.0	1.75	0.35
Y5	EVD	5.0	1.75	0.35

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	2.945	2.959 (2.959)	0.47
	Pf	1.615 E-3	1.545 E-3 (1.545 E-3)	4.53
Wu/FPI	β	2.866	2.877 (2.810)	0.38
	Pf	2.078 E-3	2.011 E-3 (2.477 E-3)	3.33
Monte Carlo(4)	Pf	2.140 E-3		

(1) Developed by C. Kelly, Y. T. Wu
(2) With improvements to numerical algorithms: (a) gamma function, (b) Weibull shape parameter, (c) standard normal cdf, (d) inverse normal cdf, (e) EVD parameters
(3) Assumes new program is exact
(4) Does <u>not</u> have improvements in numerical algorithms

EXAMPLE 5 Note: ABS program results in parentheses

-DATA

FAILURE FUNCTION : $g = X_1 + X_2 + X_3 + X_4 + X_5 - Y_1 - Y_2 - Y_3 - Y_4 -$

FAILURE EVENT : $g < 0$

VARIABLE	DISTRIBUTION	MEAN/MEDIAN*	STD. DEV.	C.O.V.
X_1	EVD	10.0	4.0	0.4
X_2	WEIBULL	10.0	4.0	0.4
X_3	LOGNORMAL	9.2847 *	4.0	0.4
X_4	EVD	10.0	4.0	0.4
X_5	WEIBULL	10.0	4.0	0.4
Y_1	EVD	5.0	2.0	0.4
Y_2	WEIBULL	5.0	2.0	0.4
Y_3	LOGNORMAL	4.6424 *	2.0	0.4
Y_4	EVD	5.0	2.0	0.4
Y5	WEIBULL	5.0	2.0	0.4

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE % (3)
R-F	β	2.649	2.652 (2.652)	0.11
	Pf	4.031 E-3	4.003 E-3 (4.003 E-3)	0.70
Wu/FPI	β	2.696	2.698 (2.658)	0.07
	Pf	3.508 E-3	3.491 E-3 (3.984 E-3)	0.49
Monte Carlo(4)	Pf	3.643 E-3		

(1) Developed by C. Kelly, Y. T. Wu

(2) With improvements to numerical algorithms: (a) gamma function, (b) Weibull shape parameter, (c) standard normal cdf, (d) inverse normal cdf, (e) EVD parameters

(3) Assumes new program is exact

(4) Does <u>not</u> have improvements in numerical algorithms

EXAMPLE 6 Note: ABS program results in parentheses

-DATA

FAILURE FUNCTION : $g = R - (\frac{4T}{\pi D^2})$

FAILURE EVENT : $g < 0$

VARIABLE	DISTRIBUTION	MEAN/MEDIAN*	STD. DEV.	C.O.V.
R	NORMAL	170.	25.	0.14706
D	NORMAL	29.4	3.	0.10204

T = 50,000

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	2.902	2.902 (2.902)	0.00
	Pf	1.856 E-3	1.856 E-3 (1.856 E-3)	0.00
Wu/FPI	β	2.835	2.834 (2.833)	0.03
	Pf	2.296 E-3	2.297 E-3 (2.306 E-3)	0.04
Exact solution	Pf	2.301 E-3		

(1) Developed by C. Kelly, Y. T. Wu

(2) With improvements to numerical algorithms: (a) gamma function, (b) Weibull shape parameter, (c) standard normal cdf, (d) inverse normal cdf, (e) EVD parameters

(3) Assumes new program is exact

T = 20,000

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	5.273	5.273	0.00
	Pf	0.673 E-7	0.673 E-7	0.00
Wu/FPI	β	5.111	5.110	0.02
	Pf	1.599 E-7	1.612 E-7	0.81
Exact solution	Pf	1.502 E-7		

T = 5,000

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	6.492	6.492	0.00
	Pf	4.238 E-11	4.236 E-11	0.005
Wu/FPI	β	6.484	6.484	0.00
	Pf	4.453 E-11	4.459 E-11	0.02
Exact solution	Pf	4.646 E-11		

EXAMPLE 7 Note: ABS program results in parentheses

-DATA

FAILURE FUNCTION : $g = R - \sqrt{300P^2 + 1.92T^2}$

FAILURE EVENT : $g < 0$

VARIABLE	DISTRIBUTION	MEAN/MEDIAN*	STD. DEV.	C.O.V.
R	WEIBULL	48.0	3.0	0.0625
P	LOGNORMAL	0.987 *	0.16	0.16
T	EVD	20.0	2.0	0.1

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	3.094	3.085 (3.085)	0.29
	Pf	0.988 E-3	1.018 E-3 (1.016 E-3)	2.95
Wu/FPI	β	2.893	2.886 (2.868)	0.24
	Pf	1.911 E-3	1.950 E-3 (2.064 E-3)	2.0
Monte Carlo(4)	Pf	1.800 E-3		

(1) Developed by C. Kelly, Y. T. Wu

(2) With improvements to numerical algorithms: (a) gamma function, (b) Weibull shape parameter, (c) standard normal cdf, (d) inverse normal cdf, (e) EVD parameters

(3) Assumes new program is exact

(4) Does <u>not</u> have improvements in numerical algorithms

EXAMPLE 8

-DATA

FAILURE FUNCTION : $g = K - S\sqrt{\pi A}$

FAILURE EVENT : $g < 0$

VARIABLE	DISTRIBUTION	MEAN/MEDIAN*	STD. DEV.	C.O.V.
K	WEIBULL	150.	25.0	0.16667
S	EVD	100.	20.0	0.2
A	LOGNORMAL	0.1 *	0.1414	1.0

$\mu_s = 100$
$\sigma_s = 20$

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	2.060	2.067	0.34
	Pf	1.968 E-2	1.938 E-2	1.55
Wu/FPI	β	1.967	1.974	0.35
	Pf	2.461 E-2	2.419 E-2	1.74
Monte Carlo(4)	Pf	2.412 E-2		

(1) Developed by C. Kelly, Y. T. Wu
(2) With improvements to numerical algorithms: (a) gamma function, (b) Weibull shape parameter, (c) standard normal cdf, (d) inverse normal cdf, (e) EVD parameters
(3) Assumes new program is exact
(4) Does <u>not</u> have improvements in numerical algorithms

EXAMPLE 8

-DATA

FAILURE FUNCTION : $g = K - S\sqrt{\pi A}$

FAILURE EVENT : $g < 0$

VARIABLE	DISTRIBUTION	MEAN/MEDIAN*	STD. DEV.	C.O.V.
K	WEIBULL	150.	25.0	0.16667
S	EVD	80.	16.0	0.2
A	LOGNORMAL	0.1 *	0.1414	1.0

$\mu_s = 80$
$\sigma_s = 16$

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	2.482	2.490	0.32
	Pf	6.534 E-3	6.382 E-3	2.38
Wu/FPI	β	2.380	2.389	0.38
	Pf	8.672 E-3	8.453 E-3	2.59
Monte Carlo(4)	Pf	8.630 E-3		

(1) Developed by C. Kelly, Y. T. Wu
(2) With improvements to numerical algorithms: (a) gamma function, (b) Weibull shape parameter, (c) standard normal cdf, (d) inverse normal cdf, (e) EVD parameters
(3) Assumes new program is exact
(4) Does <u>not</u> have improvements in numerical algorithms

EXAMPLE 8 Note: ABS program results in parentheses

-DATA

FAILURE FUNCTION : $g = K - S\sqrt{\pi A}$

FAILURE EVENT : $g < 0$

VARIABLE	DISTRIBUTION	MEAN/MEDIAN*	STD. DEV.	C.O.V.
K	WEIBULL	150.	25.0	0.16667
S	EVD	60.	12.0	0.2
A	LOGNORMAL	0.1 *	0.1414	1.0

$\mu_s = 60$

$\sigma_s = 12$

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	3.006	3.018 (3.018)	0.40
	Pf	1.323 E-3	1.272 E-3 (1.272 E-3)	4.01
Wu/FPI	β	2.892	2.905 (2.897)	0.45
	Pf	1.914 E-3	1.835 E-3 (1.886 E-3)	4.30
Monte Carlo(4)	Pf	1.870 E-3		

(1) Developed by C. Kelly, Y. T. Wu

(2) With improvements to numerical algorithms: (a) gamma function, (b) Weibull shape parameter, (c) standard normal cdf, (d) inverse normal cdf, (e) EVD parameters

(3) Assumes new program is exact

(4) Does <u>not</u> have improvements in numerical algorithms

EXAMPLE 9 Note: ABS program results in parentheses

-DATA

FAILURE FUNCTION : $g = \Delta - N_0 \left\{ \frac{fpp}{G\,(Y\,\Delta\varepsilon_0)^{Y}} + \frac{1 - fpp}{H\,(Y\,\Delta\varepsilon_0)^{\eta}} \right\}$

FAILURE EVENT : $g < 0$

VARIABLE	DISTRIBUTION	MEAN/MEDIAN*	STD. DEV.	C.O.V.
Δ	LOGNORMAL	1.0 *	0.3132	0.3
fpp	NORMAL	0.7	0.07	0.1
G	LOGNORMAL	0.222 *	0.0956	0.4
Y	LOGNORMAL	1.0 *	0.1517	0.15
$\Delta\varepsilon_0$	EVD	0.0005	0.00008	0.16
H	LOGNORMAL	1.673 *	0.7208	0.4

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE % (3)
R-F	β	2.384	2.385 (2.385)	0.04
	Pf	8.552 E-3	8.550 E-3 (8.550 E-3)	0.02
Wu/FPI	β	2.338	2.338 (2.315)	0.00
	Pf	9.696 E-3	9.691 E-3 (10.320 E-3)	0.05
Monte Carlo(4)	Pf	10.020 E-3		

(1) Developed by C. Kelly, Y. T. Wu
(2) With improvements to numerical algorithms: (a) gamma function, (b) Weibull shape parameter, (c) standard normal cdf, (d) inverse normal cdf, (e) EVD parameters
(3) Assumes new program is exact
(4) Does <u>not</u> have improvements in numerical algorithms

EXAMPLE 10

-DATA

FAILURE FUNCTION : $g = R^2 - X_1 - X_2$

FAILURE EVENT : $g < 0$

VARIABLES	Xi , i = 1 , 2 .
DISTRIBUTION	All Xi are Chi-Square distribution with degree of freedom v = 1 .
MEAN	1.0
STD. DEV.	1.4142
C. O. V.	1.4142
CONSTANT , R	3 , 4 , 5 .

R = 3

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	2.584	2.583	0.04
	Pf	0.489 E-2	0.490 E-2	0.20
Wu/FPI	β	2.178	2.178	0.00
	Pf	1.471 E-2	1.471 E-2	0.00
Exact solution	Pf	1.110 E-2		

(1) Developed by C. Kelly, Y. T. Wu
(2) With improvements to numerical algorithms: (a) gamma function, (b) Weibull shape parameter, (c) standard normal cdf, (d) inverse normal cdf, (e) EVD parameters
(3) Assumes new program is exact

R = 4

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	3.676	3.675	0.03
	Pf	1.186 E-4	1.189 E-4	0.08
Wu/FPI	β	3.393	3.390	0.09
	Pf	3.456 E-4	3.494 E-4	1.09
Monte Carlo(4)	Pf	3.350 E-4		

R = 5

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	4.735	4.735	0.00
	Pf	1.096 E-6	1.098 E-6	0.18
Wu/FPI	β	4.545	4.535	0.22
	Pf	2.745 E-6	2.879 E-6	4.65
Exact solution	Pf	3.730 E-6		

EXAMPLE 10

-DATA

FAILURE FUNCTION : $g = R^2 - X_1 - X_2 - X_3 - X_4 - X_5$

FAILURE EVENT : $g < 0$

VARIABLES	Xi , i = 1, 2, 3, 4, 5.
DISTRIBUTION	All Xi are Chi-Square distribution with degree of freedom v = 1 .
MEAN	1.0
STD. DEV.	1.4142
C. O. V.	1.4142
CONSTANT , R	3 , 4 , 5 .

-COMPARISONS OF SAFETY INDEX AND PROBABILITY OF FAILURE , Pf

R = 3

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	2.049	2.049	0.00
	Pf	2.023 E-2	2.022 E-2	0.05
Wu/FPI	β	1.302	1.301	0.08
	Pf	9.652 E-2	9.655 E-2	0.03
Exact solution	Pf	1.090 E-;		

R = 4

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	3.241	3.241	0.00
	Pf	5.954 E-4	5.966 E-4	0.20
Wu/FPI	β	2.447	2.447	0.00
	Pf	7.220 E-3	7.195 E-3	0.38
Exact solution	Pf	6.840 E-3		

R = 5

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	4.380	4.380	0.00
	Pf	5.930 E-6	5.951 E-6	0.35
Wu/FPI	β	3.574	3.578	0.11
	Pf	1.761 E-4	1.733 E-4	1.62
Exact solution	Pf	1.390 E-4		

EXAMPLE 10

-DATA

FAILURE FUNCTION : $g = R^2 - X_1 - X_2 - X_3 - X_4 - X_5 - X_6 - X_7 - X_8 - X_9 - X_{10}$

FAILURE EVENT : $g < 0$

VARIABLES	Xi , i = 1,2,3,4,5,6,7,8,9,10.
DISTRIBUTION	All Xi are Chi-Square distribution with degree of freedom v = 1 .
MEAN	1.0
STD. DEV.	1.4142
C. O. V.	1.4142
CONSTANT , R	4 , 5 , 6 .

-COMPARISONS OF SAFETY INDEX AND PROBABILITY OF FAILURE , Pf

R = 4

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	2.595	2.595	0.00
	Pf	4.733 E-3	4.725 E-3	0.17
Wu/FPI	β	1.254	1.254	0.00
	Pf	1.049 E-1	1.050 E-1	0.09
Exact solution	Pf	0.060 E-2		

R = 5

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	3.815	3.815	0.00
	Pf	6.808 E-5	6.819 E-5	0.16
Wu/FPI	β	2.749	2.750	0.04
	Pf	2.988 E-3	2.984 E-3	0.13
Exact solution	Pf	5.350 E-3		

R = 6

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	4.977	4.976	0.02
	Pf	3.266 E-7	3.243 E-7	0.71
Wu/FPI	β	3.812	3.815	0.08
	Pf	6.885 E-5	6.818 E-5	0.98
Exact solution	Pf	8.420 E-5		

EXAMPLE 10

-DATA

FAILURE FUNCTION : $g = R^2 - X_1 - X_2 - X_3 - X_4 - X_5 - X_6 - X_7 - X_8 - X_9 - X_{10} - X_{11} - X_{12} - X_{13} - X_{14} - X_{15} - X_{16} - X_{17} - X_{18} - X_{19} - X_{20}$

FAILURE EVENT : $g < 0$

VARIABLES	Xi , i = 1,2,3,...,18,19,20.
DISTRIBUTION	All Xi are Chi-Square distribution with degree of freedom v = 1 .
MEAN	1.0
STD. DEV.	1.4142
C. O. V.	1.4142
CONSTANT , R	5 , 6 , 7 , 8 .

-COMPARISONS OF SAFETY INDEX AND PROBABILITY OF FAILURE , Pf

R = 5

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	2.827	2.828	0.04
	Pf	2.351 E-3	2.340 E-3	0.27
Wu/FPI	β	0.441	0.440	0.23
	Pf	3.293 E-1	3.300 E-1	0.21
Exact solution	Pf	2.010 E-1		

R = 6

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	4.098	4.099	0.02
	Pf	2.084 E-5	2.080 E-5	0.19
Wu/FPI	β	2.122	2.121	0.05
	Pf	1.692 E-2	1.694 E-2	0.12
Exact solution	Pf	1.540 E-2		

R = 7

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	5.311	5.310	0.02
	Pf	5.464 E-8	5.479 E-8	0.27
Wu/FPI	β	3.370	3.371	0.03
	Pf	3.755 E-4	3.744 E-4	0.29
Exact solution	Pf	3.070 E-4		

R = 8

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	6.482	6.481	0.02
	Pf	4.517 E-11	4.553 E-11	0.81
Wu/FPI	β	4.502	4.505	0.07
	Pf	3.365 E-6	3.320 E-6	1.4
Exact solution	Pf	1.680 E-6		

EXAMPLE 11

-DATA

FAILURE FUNCTION : $g = 2.5 - N \frac{Cc}{1 + e_0} H \log \frac{P_0 + \Delta P}{P_0}$

FAILURE EVENT : $g < 0$

VARIABLE	DISTRIBUTION	MEAN/MEDIAN*	STD. DEV.	C.O.V.
N	NORMAL	1.0	0.1	0.10
Cc	NORMAL	0.396	0.099	0.25
e_0	NORMAL	1.19	0.1785	0.15
H	NORMAL	168.0	8.4	0.05
P_0	NORMAL	3.72	0.186	0.05
ΔP	NORMAL	0.35	0.07	0.20

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	2.439	2.439	0.00
	Pf	7.363 E-3	7.363 E-3	0.00
Wu/FPI	β	2.499	2.499	0.00
	Pf	6.235 E-3	6.229 E-3	0.10
Monte Carlo(4)	Pf	6.330 E-3		

(1) Developed by C. Kelly, Y. T. Wu
(2) With improvements to numerical algorithms: (a) gamma function, (b) Weibull shape parameter, (c) standard normal cdf, (d) inverse normal cdf, (e) EVD parameters
(3) Assumes new program is exact
(4) Does <u>not</u> have improvements in numerical algorithms

EXAMPLE 12

-DATA

FAILURE FUNCTION : $g = N\ C\ L\ H^{3/2} - R\ Q_I$

FAILURE EVENT : $g < 0$

VARIABLE	DISTRIBUTION	MEAN/MEDIAN*	STD.DEV.	C.O.V.
N	NORMAL	1.0	0.2	0.20
C	NORMAL	3.85	0.2695	0.07
L	NORMAL	93.4	5.604	0.06
H	NORMAL	15.0	0.9	0.06
R	NORMAL	0.7	0.098	0.14
Q_I	EVD	9146.0	3201.1	0.35

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	2.715	2.715	0.00
	Pf	3.309 E-3	3.315 E-3	0.18
Wu/FPI	β	2.651	2.651	0.00
	Pf	4.019 E-3	4.017 E-3	0.05
Monte Carlo(4)	Pf	4.043 E-3		

(1) Developed by C. Kelly, Y. T. Wu
(2) With improvements to numerical algorithms: (a) gamma function, (b) Weibull shape parameter, (c) standard normal cdf, (d) inverse normal cdf, (e) EVD parameters
(3) Assumes new program is exact
(4) Does <u>not</u> have improvements in numerical algorithms

EXAMPLE 13

-DATA

FAILURE FUNCTION : $g = A + B X R^3 + C Y S^3 + D Z Q^3$

FAILURE EVENT : $g < 0$

VARIABLE	DISTRIBUTION	MEAN/MEDIAN*	STD. DEV.	C.O.V.
X	WEIBULL	10.0	3.0	0.30
Y	EVD	5.0	1.5	0.30
Z	LOGNORMAL	9.5782 *	3.0	0.30
R	EVD	10.0	3.0	0.30
S	LOGNORMAL	4.7891 *	1.5	0.30
Q	WEIBULL	10.0	3.0	0.30

$\alpha = 3$
$\beta = 3$
$\gamma = 3$

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	2.625	2.631	0.23
	Pf	4.327 E-3	4.252 E-3	1.76
Wu/FPI	β	2.720	2.724	0.15
	Pf	3.269 E-3	3.223 E-3	1.43
Monte Carlo(4)	Pf	3.357 E-3		

(1) Developed by C. Kelly, Y. T. Wu
(2) With improvements to numerical algorithms: (a) gamma function, (b) Weibull shape parameter, (c) standard normal cdf, (d) inverse normal cdf, (e) EVD parameters
(3) Assumes new program is exact
(4) Does <u>not</u> have improvements in numerical algorithms

EXAMPLE 13

-DATA

FAILURE FUNCTION : $g = A + B X R^3 + C Y S^4 + D Z Q^5$

FAILURE EVENT : $g < 0$

VARIABLE	DISTRIBUTION	MEAN/MEDIAN*	STD. DEV.	C.O.V.
X	WEIBULL	10.0	3.0	0.30
Y	EVD	5.0	1.5	0.30
Z	LOGNORMAL	9.5782 *	3.0	0.30
R	EVD	10.0	3.0	0.30
S	LOGNORMAL	4.7891 *	1.5	0.30
Q	WEIBULL	10.0	3.0	0.30

$\alpha = 3$
$\beta = 4$
$\gamma = 5$

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE z(3)
R-F	β	2.290	2.290	0.00
	Pf	1.102 E-2	1.102 E-2	0.00
Wu/FPI	β	2.410	2.422	0.50
	Pf	7.983 E-3	7.720 E-3	3.41
Monte Carlo(4)	Pf	8.020 E-3		

(1) Developed by C. Kelly, Y. T. Wu
(2) With improvements to numerical algorithms: (a) gamma function, (b) Weibull shape parameter, (c) standard normal cdf, (d) inverse normal cdf, (e) EVD parameters
(3) Assumes new program is exact
(4) Does <u>not</u> have improvements in numerical algorithms

EXAMPLE 13

-DATA

FAILURE FUNCTION : $g = A + B X R^5 + C Y S^5 + D Z Q^5$

FAILURE EVENT : $g < 0$

VARIABLE	DISTRIBUTION	MEAN/MEDIAN*	STD. DEV.	C.O.V.
X	WEIBULL	10.0	3.0	0.30
Y	EVD	5.0	1.5	0.30
Z	LOGNORMAL	9.5782 *	3.0	0.30
R	EVD	10.0	3.0	0.30
S	LOGNORMAL	4.7891 *	1.5	0.30
Q	WEIBULL	10.0	3.0	0.30

$\alpha = 5$
$\beta = 5$
$\gamma = 5$

		ORIGINAL PROGRAM(1)	NEW PROGRAM(2)	DIFFERENCE %(3)
R-F	β	2.388	2.392	0.17
	Pf	8.478 E-3	8.369 E-3	1.30
Wu/FPI	β	2.546	2.549	0.12
	Pf	5.451 E-3	5.405 E-3	0.85
Monte Carlo(4)	Pf	5.776 E-3		

(1) Developed by C. Kelly, Y. T. Wu
(2) With improvements to numerical algorithms: (a) gamma function, (b) Weibull shape parameter, (c) standard normal cdf, (d) inverse normal cdf, (e) EVD parameters
(3) Assumes new program is exact
(4) Does not have improvements in numerical algorithms

APPENDIX D

Error (or Confidence) Bounds for Distribution Functions Resulting from Statistical Sampling Error

Paul H. Wirsching

The University of Arizona

1.0 INTRODUCTION

1.1 Some Definitions and Preliminary Remarks

Let Y denote the response variable. Assume that Y will be a function of the random vector $\underset{\sim}{X}$ of design factors

where
$$Y = f(\underset{\sim}{X}) \tag{1.1}$$

$$\underset{\sim}{X} = (X_1, X_2, \ldots X_K)$$

This function is explicit and defined only through the data base generated by NESSUS,

$$(Y_i; \underset{\sim}{X}_i) \qquad i = 1, J \tag{1.2}$$

where J is the number of solution points.

It will be assumed that the basic statistical parameters for each X_i will be the mean and standard deviation, denoted as,

$$E(X_i) = \mu_i \qquad V(X_i) = \sigma_i^2 \tag{1.3}$$

The vector parameter for X_i is defined as,

$$\Theta_i = (\mu_i, \sigma_i) \tag{1.4}$$

And the parameter for $\underset{\sim}{X}$ is a vector of K elements Θ_i, for a total of 2K statistical parameters

$$\theta = (\theta_1, \theta_2, \ldots \theta_K) \tag{1.5}$$

As input to FPI, the statistical distributions of each X_i must be specified. This includes the values of θ_i.

Consider a <u>random sample</u> of X_i, $X_{i,j}$; j = 1,n. The <u>sample size</u> is n. The estimators of μ_i and σ_i^2 are,

$$\hat{\mu}_i = \frac{1}{n} \sum_{j=1}^{n} X_{i,j} \tag{1.6}$$

$$\hat{\sigma}_i^2 = \frac{1}{n-1} \sum_{j=1}^{n} (X_{i,j} - \hat{\mu}_i)^2 \tag{1.7}$$

The estimated parameters for X_i are

$$\hat{\theta}_i = (\hat{\mu}_i, \hat{\sigma}_i) \tag{1.8}$$

and for all $\underset{\sim}{X}$,

$$\hat{\theta} = (\hat{\theta}_1, \hat{\theta}_2, \ldots \hat{\theta}_K) \tag{1.9}$$

Using $\hat{\Theta}$, FPI constructs the distribution function (cdf) of Y, $\hat{F}_Y(y, \hat{\Theta})$. This is an estimate of the underlying cdf $F_Y(y)$, . . . the function which nature has chosen. An illustration of $\hat{F}_Y$ is provided in Fig. 1.1.

The distribution parameters $\hat{\Theta}$ used to construct $\hat{F}_Y$ are based on random samples. But the estimators $\hat{\Theta}$ are random variables themselves. There is uncertainty in the parameters which is reflected in $\hat{F}_Y$. This uncertainty can be described by error bounds (or confidence intervals) as illustated in Fig. 1.1. It is the goal of this analysis to develop an operational procedure for efficient estimation of these error bounds for implimentation in FPI.

In classical statistics, θ is considered to be chosen by nature and is a real number whose value remains forever unknown. The estimators $\hat{\theta}$ are

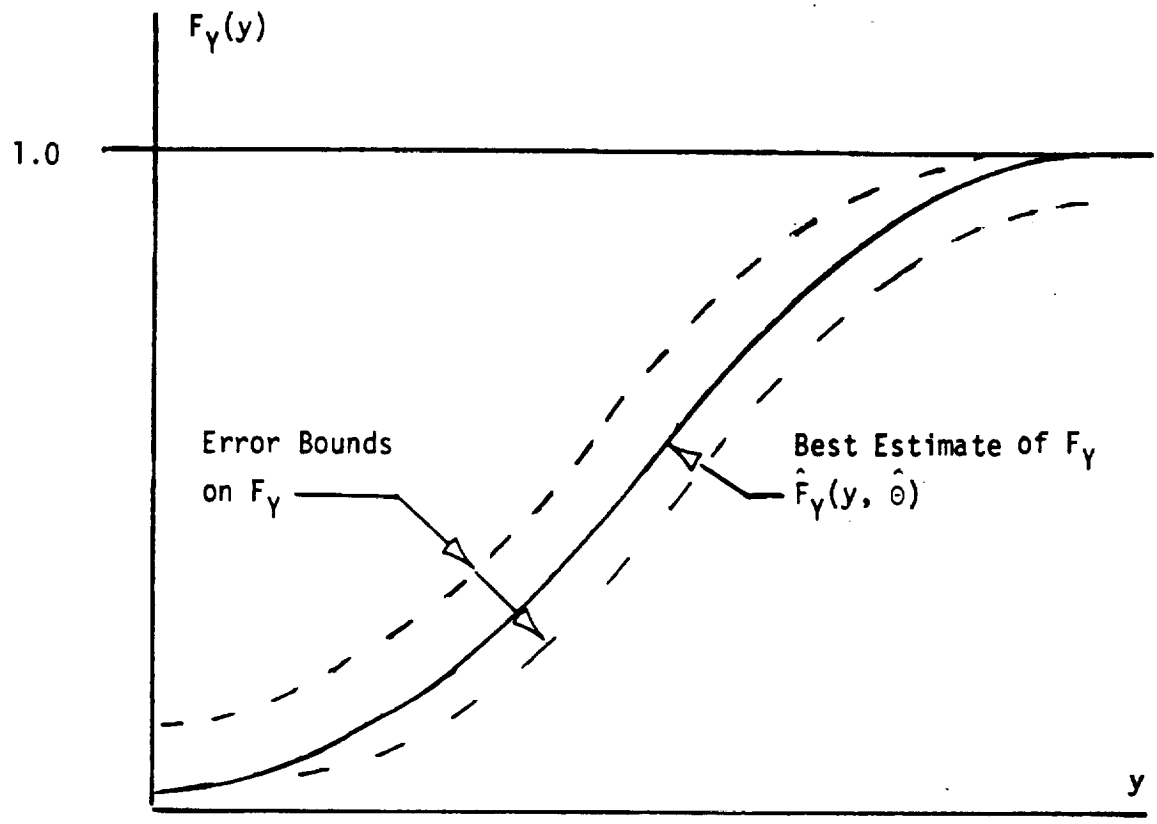


Fig. 1.1 The distribution function of response variable Y as computed by NESSUS/FPI

random variables and are used for point estimates of θ and for contructing confidence intervals on θ. But for ease of analysis of confidence bounds, it is often convenient to use a "Bayesian approach" in which θ is considered as a random variable reflecting the fact that its value is uncertain. Continuing this role reversal the estimators $\hat{\Theta}$ are assumed to be constant, and equal to the expected values of Θ. The value of this approach lies in the fact that if one can establish the distribution of Θ, then upper and lower confidence bounds are just the appropriate percentage points.

As an example, let the mean μ of a normal variate be a random variable having a mean of $\hat{\mu}$ and standard deviation of $\hat{\sigma}/\sqrt{u'}$. A direct computation of the upper 95% and lower 5% points produces the 90% confidence interval on μ. While deviating from classical statistics, this approach has experienced increased popularity in recent years.

1.2 Statement of the Problem

Consider again the cdf of Y as shown in Fig. 1.2. If $\hat{\Theta}$ were the actual values of Θ which nature has chosen, then we have perfect knowledge of the inherent variability of Y. F_Y would define precisely the distribution of Y. But if Θ is a random variable, then for a given F_Y, say F_Y', there will be uncertainty in the value of Y which produces F'. Thus, Y will be a random variable.

It is important to note that what is really wanted is not the uncertainty of Y given F_Y, but rather that of F_Y for a given Y, say y'. Thus, the general goal of this analysis will be to develop a practical algorithm for computing the error (or confidence) bounds on F_Y for a given Y = y'.

1.3 Response Variable as a Function of the Parameters

To define the distribution of Y it is necessary to have an explicit expression for Y in terms of X. This function is constructed from the data base (Eq. 1.2) as a polynomial.

$$Y = f(X) = a_o + \sum_{i=1}^{K} a_i X_i + \sum_{i=1}^{K} b_i X_i^2 + \sum_{\substack{i,j \\ i \neq j}} c_i X_i X_j \qquad (1.10)$$

Now consider the distribution of X_i, defined by the cdf, $F_i(x; \Theta_i)$ and shown in Fig. 1.3. Let X_i^* be the design point value corresponding to y'. FPI computes a design point $\underset{\sim}{X}^*$ when computing $F_Y(y')$, and in fact, must satisfy,

$$y' = f(\underset{\sim}{X}^*) \qquad (1.11)$$

The cdf corresponding to X_i^* is denoted as F_i^*. At F_i^*, X_i can be written in terms of Θ_i by inverting the cdf

$$X_i(\Theta_i) = F_i^{-1}(F_i^*) \qquad (1.12)$$

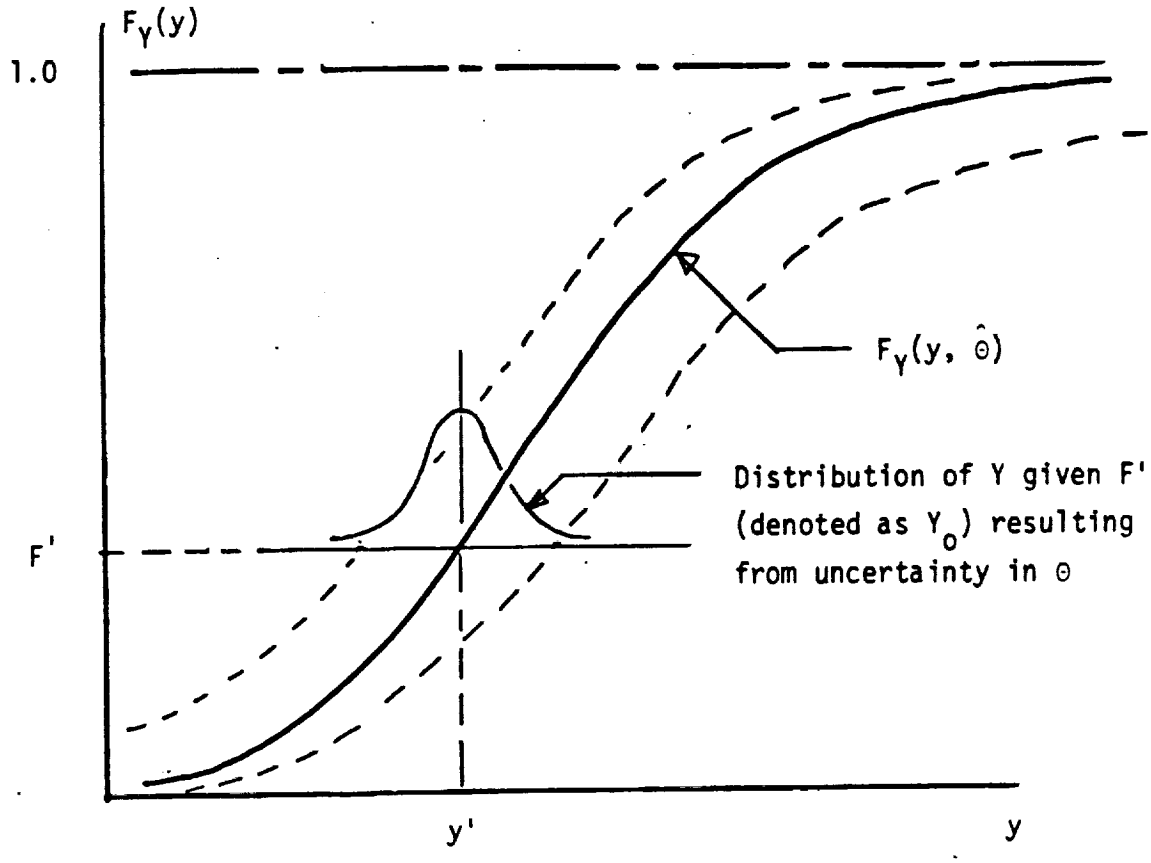


Fig. 1.2 Distribution of Y for a given F'

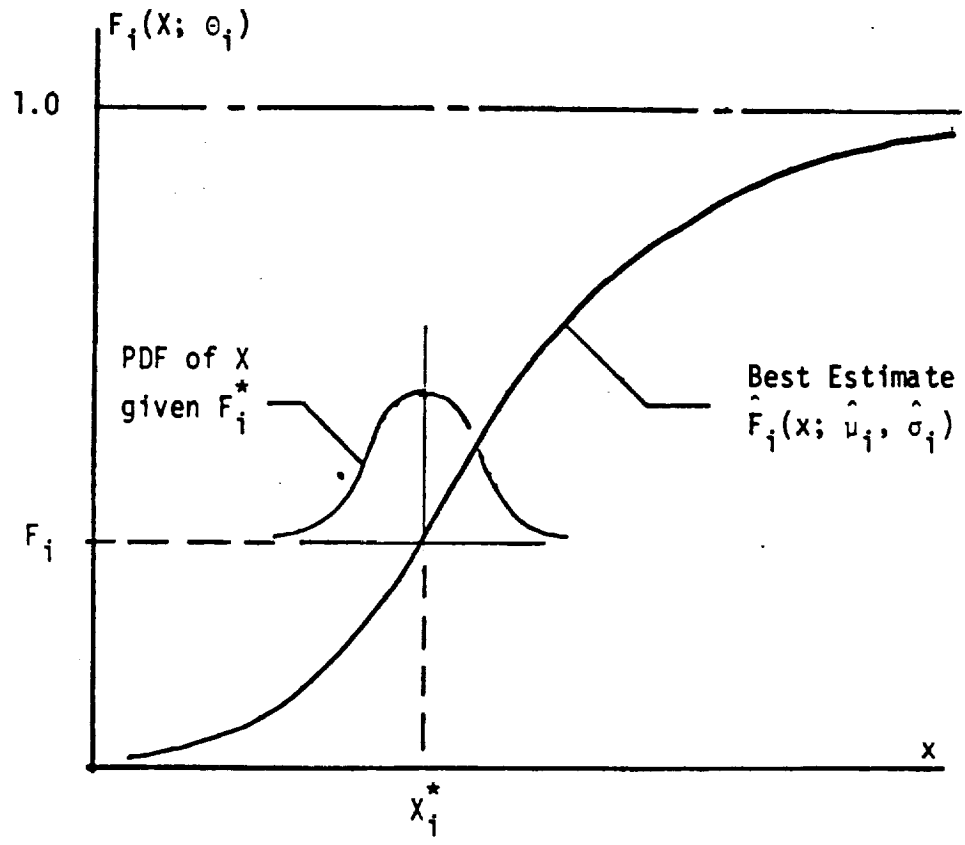


Fig. 1.3. The cdf of X_i and the pdf of X_i given F_i^* resulting from uncertainty in Θ_i

Upon substitution of each X_i into Eq. 1.10, Y can now be expressed in terms of

$$Y = g(\Theta) \tag{1.13}$$

1.4 Distribution of the Response Variable at a Given F_Y

Because Θ is a random variable, Y is a random variable. And because the uncertainty of each X_i was derived at F_i^* (and X_i^*), it follows that Eq. 1.10 defines the distribution of Y at F'. Let Y_o denote the random variable, Y at F'. The mean of Y_o should be "close to" y'.

Let the cdf and pdf of Y_o be denoted as F_o and f_o respectively; and let the mean and standard deviation of Y_o be μ_o and σ_o.

$$\mu_i(\Theta) = E(Y) = E[g(\Theta)] = y' \tag{1.14}$$

$$\sigma_o^2(\Theta) = V(Y) = V[g(\Theta)] \tag{1.15}$$

1.5 Confidence Bounds on Y_o

Let α denote the confidence level, and let y_L and y_U denote the upper and lower confidence bounds. These terms are related by the probability expression,

$$P[y_L \leq Y_o \leq y_U] = \alpha$$

And it follows that,

$$P[Y_o \leq y_L] = F_o(y_L;\, y',\, C_o) = \frac{1 - \alpha}{2}$$

$$P[Y_o \leq y_U] = F_o(y_U;\, y',\, C_o) = \frac{1 + \alpha}{2}$$

The upper and lower bounds are

$$y_L = F_o^{-1}\left(\frac{1-\alpha}{2}\right)$$

$$y_U = F_o^{-1}\left(\frac{1+\alpha}{2}\right)$$

y_L and y_U define points on the confidence boundard as shown in Fig. 1.4. Translating horizontal confidence bound to a vertical bound statement,

$$P[Y_o > y_U] = P[F < F' | y_U].$$

Thus, one point on the lower confidence bound of F_Y at y_U is obtained.

Similarly,

$$P[Y_o < y_L] = P[F > F' | y_L]$$

And a point on the upper confidence bound of F_Y at y_L is defined.

In general, then the confidence boundaries would have to be constructed on a point by point basis using several values of y'.

A simpler scheme for estimating the error bounds for F_Y at y' using calculations at y' only will be presented in the next chapter.

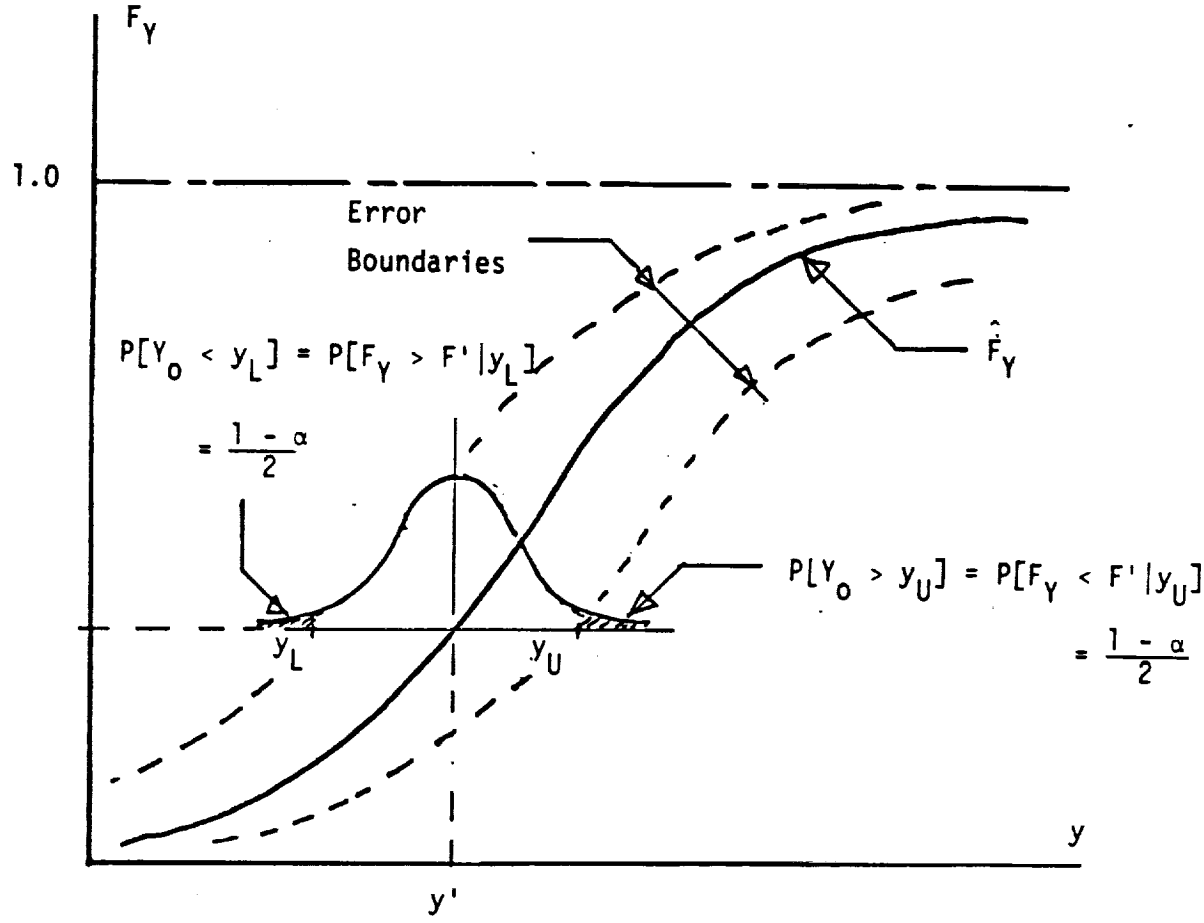


Fig. 1.4 Error bounds on Y_o and corresponding error bounds on F_Y

2.0 EXAMPLE: "FIRST ORDER" ERROR BOUNDS

2.1 Preliminary Remarks

The problem of constructing error bounds on the cdf of response variable Y, as described in Sec. 1.0, may be too general to be practical. An example provided in this section illustrates how an approximation to the error bounds can be constructed using an algorithm which is simple enough to be included without great difficulty (we hope) in a probabilistic structural code.

2.2 The Response Variable, Y

Assume that Y is linear in $\underset{\sim}{X}$ in the neighborhood of y'.

$$Y = a_o + \sum_{i=1}^{K} a_i X_i \tag{2.1}$$

The goal of the analysis is to construct the error bounds on F_Y at y'.

Assume that X_i is normal. The cdf of X_i is written as follows noting that $\Theta_i = (\mu_i, \sigma_i)$ is a random vector.

$$F_i(x;\ \mu_i, \sigma_i) = \Phi\left(\frac{x - \mu_i}{\sigma_i}\right) \tag{2.2}$$

The best estimate of the cdf of X_i is,

$$F_i^*(x;\ \mu_i, \sigma_i) = \Phi\left(\frac{x - \hat{\mu}_i}{\hat{\sigma}_i}\right) \tag{2.3}$$

Because Y is a linear function of normal X_i, the estimated distribution Y in the neighborhood of y' will be normal using the parameter estimates,

$$F_Y(y) = \Phi\left(\frac{y - \hat{\mu}_Y}{\hat{\sigma}_Y}\right) \tag{2.4}$$

where

$$\hat{\mu}_Y = a_o + \sum_{i=1}^{K} a_i \hat{\mu}_i \tag{2.5}$$

$$\hat{\sigma}_Y^2 = \sum_{i=1}^{K} a_i^2 \hat{\sigma}_i^2 \tag{2.6}$$

2.3 Properties of Y and X_i at the Design Point

A basic property of the design point values X^* used to compute $\hat{F}_Y$ at y' is

$$y' = f(\underset{\sim}{X}^*)$$

$$y' = a_o + \sum_{i=1}^{K} a_i X_i^* \tag{2.7}$$

where X_i^* is the design point associated with variable X_i.

The cdf at X_i^* is,

$$F_i^* = \Phi\left(\frac{X^* - \hat{\mu}_i}{\hat{\sigma}_i}\right) \tag{2.8}$$

Shown in Fig. 2.1 is the cdf of X_i and the point X_i^* and F_i^*.

At F_i^*, X_i can be considered as a random variable denoted as X_{oi}, because it is a function of $\Theta_i = (\mu_i, \sigma_i)$.

$$X_{oi}(\mu_i, \sigma_i) = F_i^{-1}(F_i^*) \tag{2.9}$$

$$= \sigma_i \Phi^{-1}[F_i^*] + \mu_i$$

Upon substituting Eq. 2.8, it follows that,

$$X_{oi}(\mu_i, \sigma_i) = \sigma_i x^* - \mu_i \tag{2.10}$$

where,

$$x^* = \frac{X^* - \hat{\mu}_i}{\hat{\sigma}_i} \tag{2.11}$$

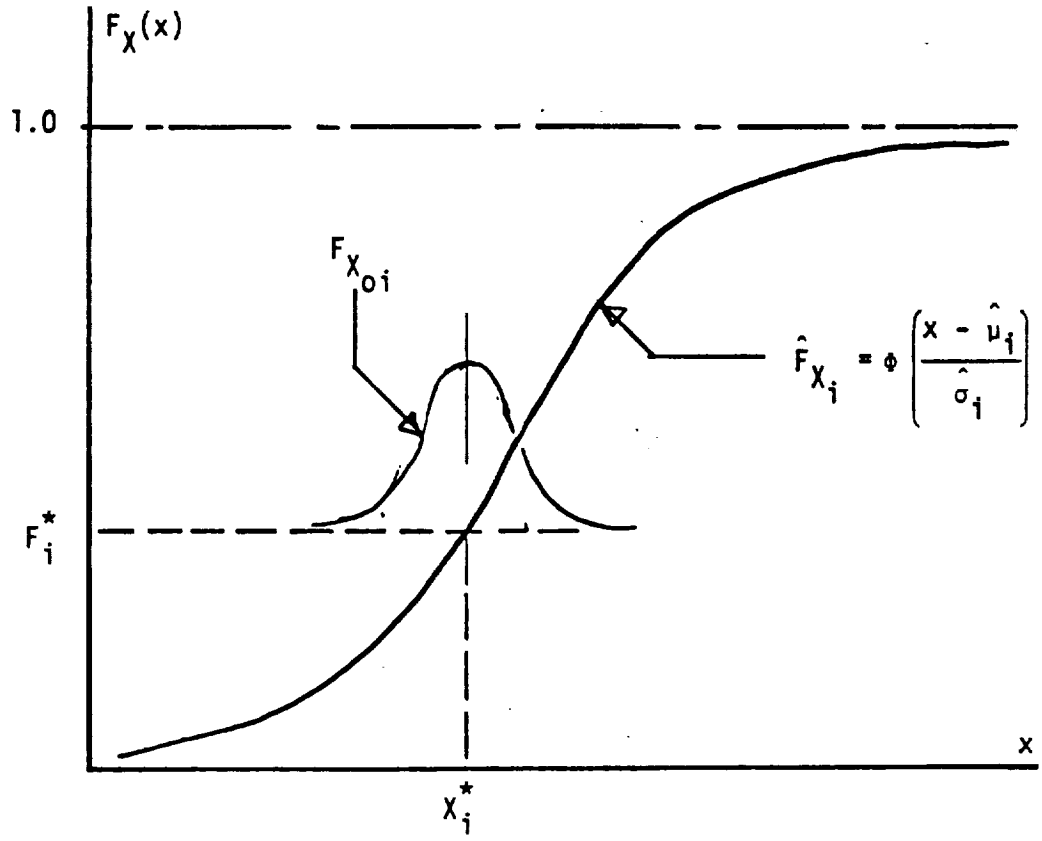


Fig. 2.1 The cdf of X_i showing the design point and corresponding F_i^*

Note that

$$E(X_{oi}) = X^* \tag{2.12}$$

because $E(\sigma_i) = \hat{\sigma}_i$, and $E(\mu_i) = \hat{\mu}_i$.

Also note that X_{oi} in Eq. 2.10 is a random variable because σ_i and μ_i are random variables. The pdf of X_{oi} is shown in Fig. 2.1.

2.4 The Distribution of the Response Variable at a Given F_Y

Define F' as the value of $\hat{F}_Y$ corresponding to y'.

$$F' = \Phi\left(\frac{y' - \hat{\mu}_Y}{\hat{\sigma}_Y}\right) \tag{2.13}$$

Define Y_o as,

$$Y_o = a_o + \sum_{i=1}^{K} a_i X_{oi} \tag{2.14}$$

$$= a_o + \sum_{i=1}^{K} a_i (\sigma_i x_i^* + \mu_i)$$

Note that from Eqs. 2.7 and 2.12,

$$E(Y_o) = y' \tag{2.15}$$

Thus Y_o is a random variable, denoting Y at F'. It is a function of Θ, and it models or represents the error bound in Y at a given F'. The distribution of Y is shown in Fig. 2.2.

The standard deviation of Y_o is

$$\sigma_o = \left\{ \sum_{i=1}^{K} a_i^2 \left[(x_i^*)^2 V(\sigma_i) + V(\mu_i) \right] \right\}^{1/2} \tag{2.16}$$

And the coefficient of variation (COV) of Y_o is

$$C_o = \sigma_o / \mu_o \tag{2.17}$$

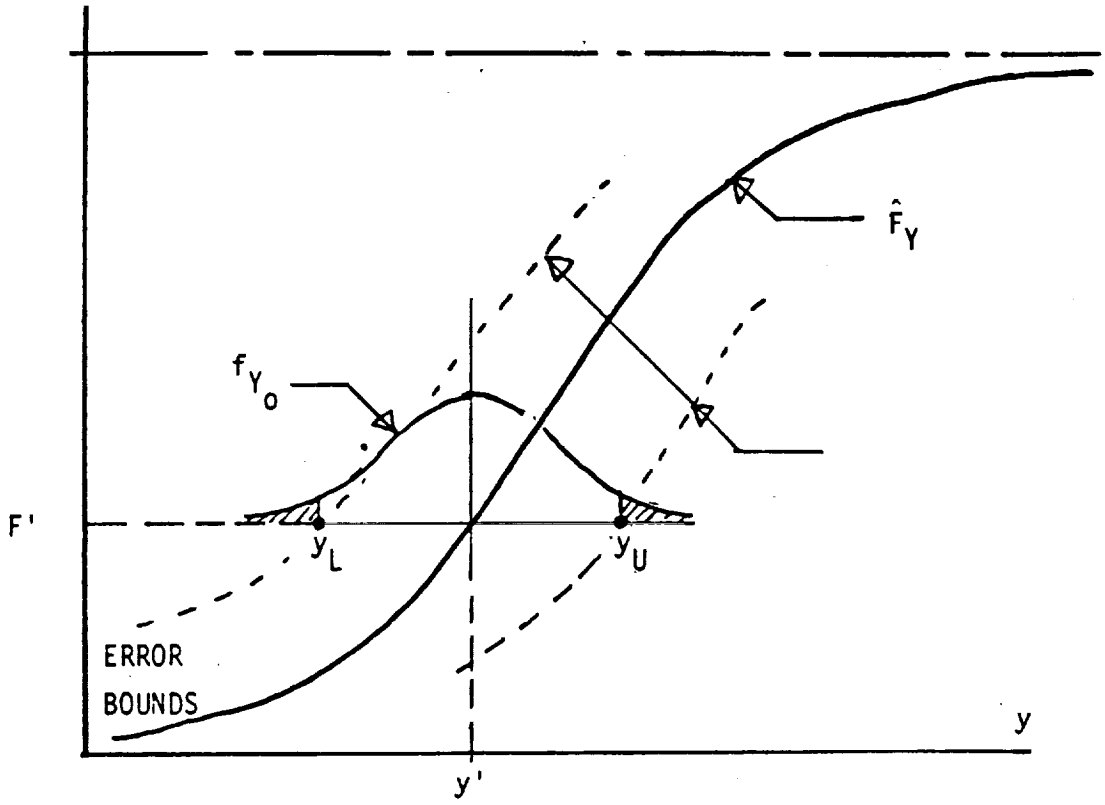


Fig. 2.2 The estimated distribution function for response variable Y and upper and lower error α error bounds for Y

If X_i is normal, then μ_i will be normal and σ_i will have an x^2 distribution. In general, it would be difficult to derive the distribution of Y_o, but as indicated below, a "default" lognormal model can be assumed. As a general purpose distribution, the lognormal can be used as an approximating model in a variety of applications in which the exact form cannot be found.

2.5 Upper and Lower Error Bounds on Y_o

Option 1. The lognormal model for Y_o. Assume that Y_o has a lognormal distribution. Upper and lower error bounds on Y_o, denoted as y_U and y_L, and shown in Fig. 2.2, can be derived as follows: Let α be the confidence level. Then, y_L, for example, is related to α as,

$$P\left[Y < y_L | F'\right] = \frac{1 - \alpha}{2} \tag{2.18}$$

And if Y_o is lognormal, it follows that the lower error bound for Y is,

$$y_L = \tilde{Y}_o \exp(z_L \delta) \tag{2.19}$$

where

$$\tilde{Y}_o = y'/\sqrt{1 + C_o^2} \tag{2.20}$$

$$\delta = \sqrt{\ln(1 + C_o^2)} \tag{2.21}$$

z_L = standard normal variate at a probability level of $(1 - \alpha)/2$

Similarly, the upper error bound for Y_o is,

$$y_U = \tilde{Y}_o \exp(z_u \delta) \tag{2.22}$$

where z_U is the standard normal variate at a probability level of $(1 + \alpha)/2$.

Option 2. The normal model for Y_o. Unfortunately, the lognormal model for Y_o has a serious limitation. The lognormal distribution is defined only for $Y > 0$. If the response variable has values at zero, or in the "neighborhood" of zero, then the lognormal is not suitable. In general, this may not be a problem, but it is not unreasonable to imagine interest in some variable which has a zero mean. In any case, the problem can be avoided by using a normal model for Y. The penalty may be a loss of accuracy.

The mean and standard deviation of Y_o are equal to y' and σ_o (Eq. 2.16) respectively. Then the upper and lower error bounds on Y_o are,

$$y_L = z_L \sigma_Y + y' \tag{2.23}$$

$$y_U = z_U \sigma_Y + y' \tag{2.24}$$

2.6 Translation of Error Bounds on Y to Error Bounds on F_Y

Assuming that $Y = f(\underset{\sim}{X})$ is linear in the neighborhood of y' and that all X_i are normal, it follows that Y will also be normal. Error bounds on Y_o can then be easily transferred to F given y'. The scheme for doing this is suggested in Fig. 2.3.

The standard normal variate z defines the estimated distribution of Y,

$$z = \frac{y - \hat{\mu}_Y}{\hat{\sigma}_Y} \tag{2.25}$$

where,

$$\hat{F}_Y(y) = \Phi(z) \tag{2.26}$$

At y' the error bounds are assumed to be parallel to F_Y as shown in Fig. 2.3. The slope of the line is

$$\frac{dz}{dX} = \frac{1}{\hat{\sigma}_Y} \tag{2.27}$$

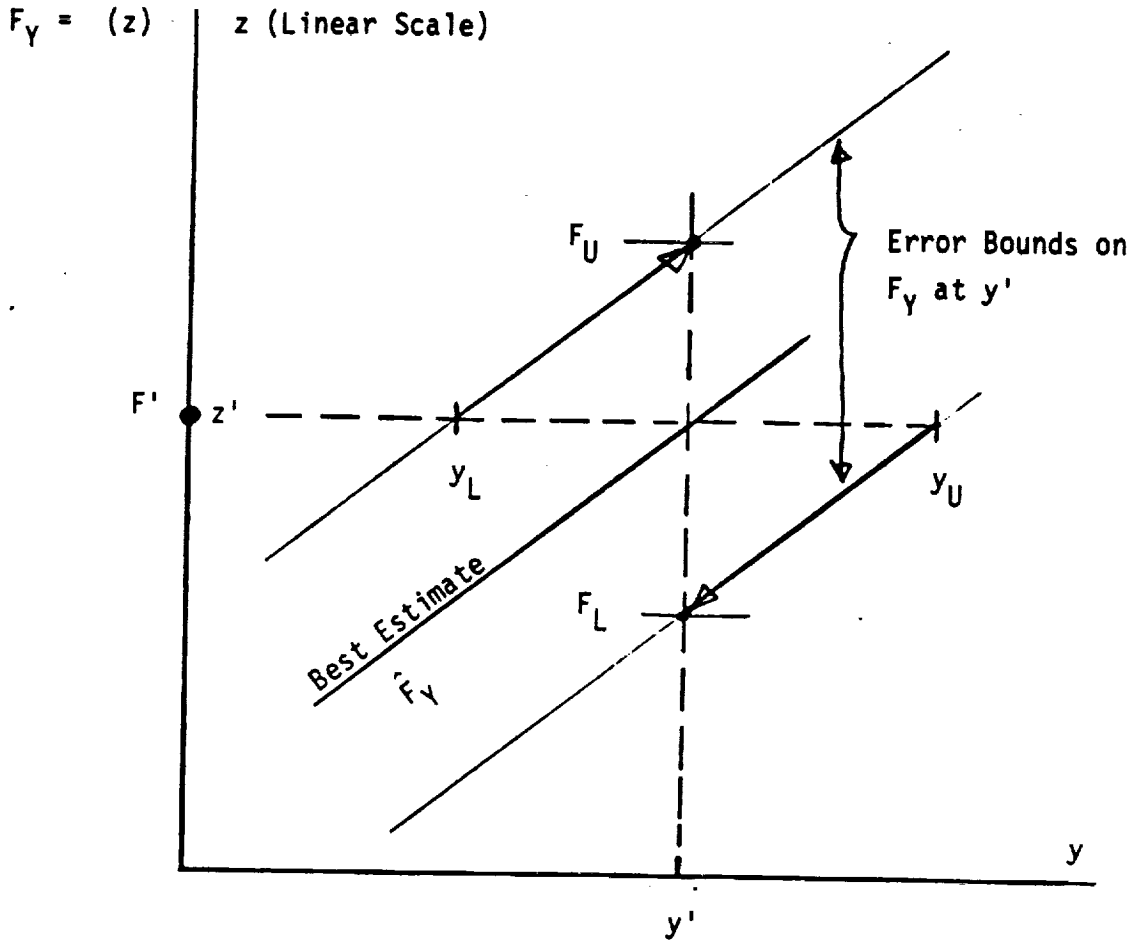


Fig. 2.3 How Error Bounds on Y are Translated to Error Bounds on F_Y

And it is seen from Fig. 2.3 that

$$F_L = \Phi\left(z' - \frac{y_U - y'}{\hat{\sigma}_Y}\right) \tag{2.28}$$

$$F_U = \Phi\left(z' + \frac{y' - y_L}{\hat{\sigma}_Y}\right) \tag{2.29}$$

where

$$z' = \frac{y' - \hat{\mu}_Y}{\hat{\sigma}_Y} \tag{2.30}$$

2.7 Concluding Remarks

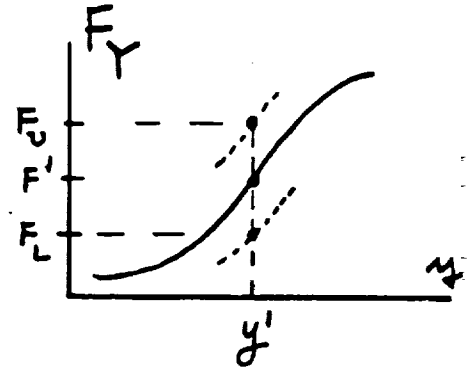
These first order error bounds were derived on the basis of distributional assumptions. It is hypothesized that these bounds are robust in that they provide a reasonable approximation to those bounds for the general case where $y = f(\underset{\sim}{X})$ is not linear, and $\underset{\sim}{X}$ is not normal. This has yet to be proven.

A numerical example is provided in the next chapter.

SUMMARY; OPTION 1 (LOGNORMAL Y_0)

$$F_L = \Phi\left(\zeta' - \frac{y_U - y'}{\hat{\sigma}_Y}\right)$$

$$F_U = \Phi\left(\zeta' + \frac{y' - y_L}{\hat{\sigma}_Y}\right)$$



WHERE,

$$\hat{\mu}_Y = a_0 + \sum_{i=1}^{k} a_i \hat{\mu}_i \qquad Y = a_0 + \sum_{i=1}^{k} a_i X_i$$

$$\hat{\sigma}_Y = \left[\sum_{i=1}^{k} a_i^2 \hat{\sigma}_i^2\right]^{1/2}$$

$$y_L = \tilde{Y}_0 \exp(\zeta_L \delta)$$

$$y_U = \tilde{Y}_0 \exp(\zeta_U \delta)$$

$$\tilde{Y}_0 = y' / \sqrt{1 + C_0^2}$$

$$\delta = \sqrt{\ln(1 + C_0^2)}$$

ζ_L = standard normal variate at a probability level of $(1-\alpha)/2$

ζ_U = " " " $(1+\alpha)/2$

$$C_0 = \sigma_0 / \mu_0$$

$$\sigma_0 = \sqrt{\sum_{i=1}^{k} a_i^2 \{(x_i^*)^2 V(\sigma_i) + V(\mu_i)\}}$$

$$\mu_o = y^1$$

$$z^1 = \frac{y^1 - \hat{\mu}_Y}{\hat{\sigma}_Y}$$

x_i^* = DESIGN POINT

$$y^1 = a_o + \sum_{i=1}^{n} a_i x_i^*$$

$$x_i^* = \frac{X_i^* - \hat{\mu}_i}{\hat{\sigma}_i}$$

FROM DATA:

SAMPLE X_i

$$\underset{\sim}{X}_i = (X_{i,1}, X_{i,2}, \cdots X_{i,n})$$

n = SAMPLE SIZE

$$\hat{\mu}_i = \bar{X}_i = \frac{1}{n} \sum_{j=1}^{n} X_{i,j}$$

$$\hat{\sigma}_i = s_i = \left[\frac{1}{n-1} \sum_{j=1}^{n} (X_{i,j} - \bar{X}_i)^2 \right]$$

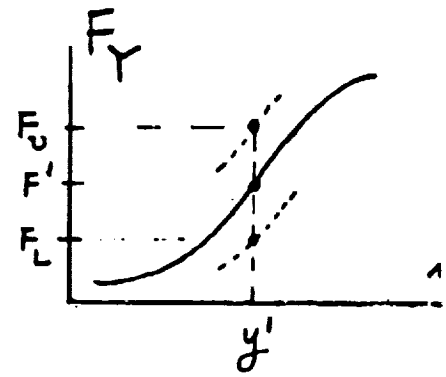
$$V(\mu_i) = s_i^2 / n$$

$$V(\sigma_i) = \frac{s_i^2}{2(n-1)}$$

SUMMARY; OPTION 2 (NORMAL Y_0)

$$F_L = \Phi\left(\zeta' - \frac{y_U - y'}{\hat{\sigma}_Y}\right)$$

$$F_U = \Phi\left(\zeta' + \frac{y' - y_L}{\hat{\sigma}_Y}\right)$$



WHERE,

$$\hat{\mu}_Y = a_0 + \sum_{i=1}^{k} a_i \hat{\mu}_i \qquad Y = a_0 + \sum_{i=1}^{k} a_i X_i$$

$$\hat{\sigma}_Y = \left[\sum_{i=1}^{k} a_i^2 \hat{\sigma}_i^2\right]^{1/2}$$

$$y_L = \zeta_L \sigma_0 + y'$$

$$y_U = \zeta_U \sigma_0 + y'$$

ζ_L = standard normal variate at a probability level of $(1-\alpha)/2$

ζ_U = " " " $(1+\alpha)/2$

$$C_0 = \sigma_0 / \mu_0$$

$$\sigma_0 = \sqrt{\sum_{i=1}^{k} a_i^2 \left\{(x_i^*)^2 V(\sigma_i) + V(\mu_i)\right\}}$$

$$\mu_o = y'$$

$$z' = \frac{y' - \hat{\mu}_Y}{\hat{\sigma}_Y}$$

x_i^* = DESIGN POINT

$$y' = a_o + \sum_{i=1}^{k} a_i x_i^*$$

$$x_i^* = \frac{X_i^* - \hat{\mu}_i}{\hat{\sigma}_i}$$

FROM DATA:

SAMPLE X_i

$$\underset{\sim}{X}_i = (X_{i,1}, X_{i,2}, \ldots X_{i,n})$$

n = SAMPLE SIZE

$$\hat{\mu}_i = \bar{X}_i = \frac{1}{n} \sum_{j=1}^{n} X_{i,j}$$

$$\hat{\sigma}_i = s_i = \left[\frac{1}{n-1} \sum_{j=1}^{n} (X_{i,j} - \bar{X}_i)^2 \right]$$

$$V(\mu_i) = s_i^2 / n$$

$$V(\sigma_i) = \frac{s_i^2}{2(n-1)}$$

3.0 EXAMPLE: A NUMERICAL EXAMPLE OF FIRST ORDER ERROR BOUNDS

Statement of the Problem

Consider the response variable Y which is a function of R and T.

$$Y = R - T$$

There is uncertainty in the parameters of R and T. It is required to compute 90% error bounds for F_Y, the distribution function of Y, at y' = 1 and y' = 2.

Observations on R and T have been made. The statistics are,

For R	For T
n = 20	n = 20
$\bar{R} = 10$	$\bar{T} = 5.0$
$s_R = 2$	$s_T = 1$

Thus the estimators are,

$$\hat{\theta}_R = (\hat{\mu}_R, \hat{\sigma}_R) \qquad \hat{\mu}_R = 10$$

$$\hat{\sigma}_R = 2$$

$$\hat{\theta}_T = (\hat{\mu}_T, \hat{\sigma}_T) \qquad \hat{\mu}_T = 5$$

$$\hat{\sigma}_T = 1$$

Solution

The calculations below follow the forms provided in the Summary The variances of the parameters are,

$$V(\mu_R) = s_R^2/n = (2)^2/20 = 0.20$$

$$V(\sigma_R) = s_R^2/2(n - 1) = (2)^2/38 = 0.105$$

and,

$$V(\mu_T) = s_T^2/n = 1/20 = 0.05$$

$$V(\sigma_T) = s_T^2/2(n - 1) = 1/38 = 0.026$$

The design points at y' = 0 and y' = 1 are defined in Fig. 3.1. Thus, at Y = y',

$$r^* = \frac{R^* - \hat{\mu}_R}{\hat{\sigma}_R} = \frac{6.8 - 10}{2} = - 1.6$$

$$t^* = \frac{T^* - \hat{\mu}_T}{\hat{\sigma}_T} = \frac{5.8 - 5}{1} = + 0.8$$

The random variable Y given F' is denoted as Y_o. The mean of Y_o is

$$\mu_o = y'$$

and the standard deviation of Y_o is,

$$\sigma_o = [(r^*)^2 V(\sigma_R) + V(\mu_R) + (t^*)^2 V(\sigma_T) + V(\mu_T)]^{1/2}$$

$$= [(1.6)^2 (1.05) + (.2) + (.18)^2 (.026) + .05]^{1/2}$$

$$\sigma_o = 0.73$$

OPTION 1 (Lognormal model for Y_o)

The COV of Y_o is,

$$C_o = \frac{\sigma_o}{\mu_o} = \frac{\sigma_o}{y'} = \frac{0.73}{1} = 0.73$$

and

$$\delta = \sqrt{\ell n(1 + C_o^2)}$$

$$= \sqrt{\ell n(1 + .73^2)} = 0.654$$

$Y = R - T$ $\qquad R \sim N(10, 2)$

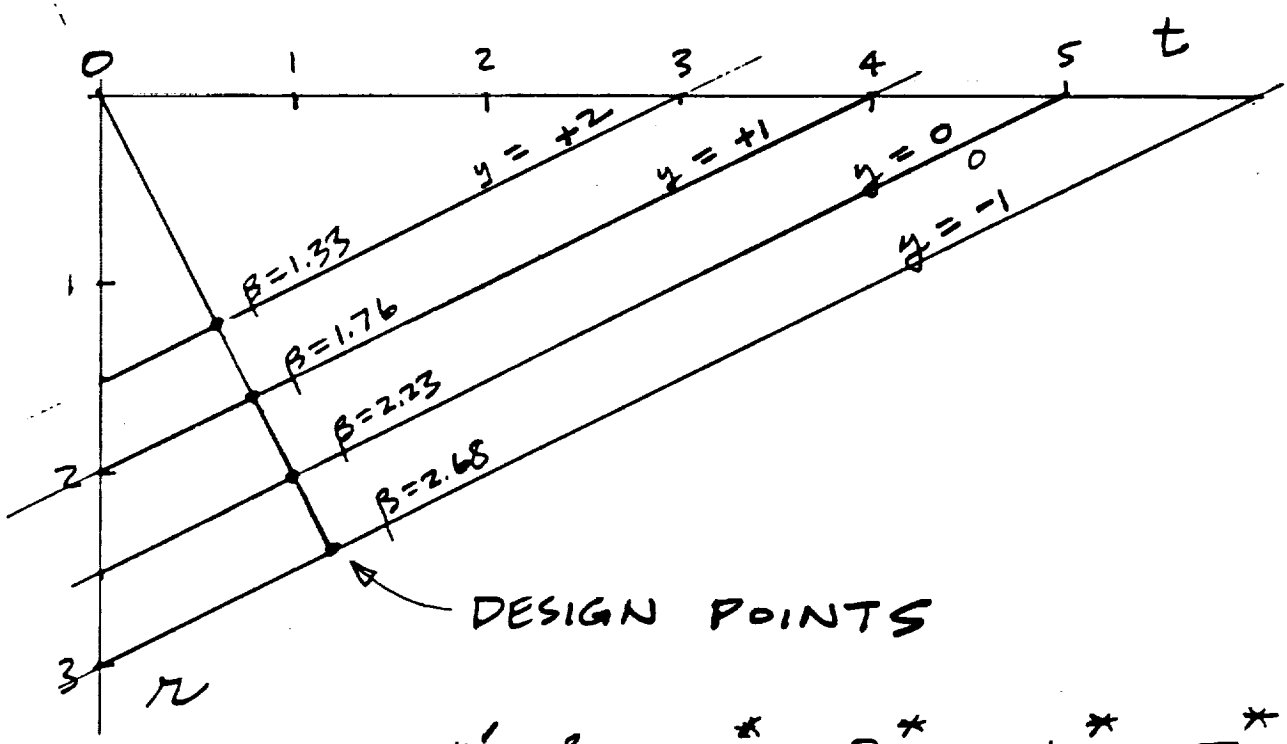
$T \sim N(5, 1)$

REDUCED VARIABLES $r = \frac{R-10}{2}$ $\quad t = \frac{T-5}{1}$

$F_Y(y) = P(Y \le y) = P(R - y - T \le 0)$

PERFORMANCE FUNCTION $g = R - y - T$

Limit State in Reduced coordinates $\qquad 2r = s - (5 - y)$



y'	β	r^*	R^*	t^*	T^*
−1	2.68	−2.4	5.2	1.2	6.2
0	2.23	−2.0	6.0	1.0	6.0
+1	1.76	−1.6	6.8	.8	5.8
+2	1.33	−1.2	7.6	.6	5.6

FIG 3.1 DESIGN POINTS CORRESPONDING TO y'.

The median of Y_o

$$\tilde{Y}_o = \frac{y'}{\sqrt{1 + C_o^2}} = \frac{1}{1.23} = 0.80$$

The estimated mean and standard deviation of Y,

$$\hat{\mu}_Y = \hat{\mu}_R - \hat{\mu}_T = 10 - 5 = 5$$

$$\hat{\sigma}_Y = \sqrt{\hat{\sigma}_R^2 + \hat{\sigma}_T^2}$$

$$= \sqrt{2^2 + 1^2} = 2.24$$

For 90% error bounds, $(1 - \alpha)/2 = 0.05$, and $(1 + \alpha)/2 = 0.95$.

$$z_L = - 1.64 \qquad z_U = + 1.64$$

The upper and lower error bounds for Y_o,

$$y_L = \tilde{Y}_o \exp(z_L \delta) = 0.80 \exp (-1.64 \times .654)$$

$$= 0.273$$

$$y_U = \tilde{Y}_o \exp(z_U \delta) = 0.80 \exp (1.64 \times .654)$$

$$= 2.33$$

The cdf of Y at $y' = 1$ is,

$$F' = \Phi(z')$$

where

$$z' = \frac{y' - \hat{\mu}_Y}{\hat{\sigma}_Y}$$

$$= \frac{1 - 5}{2.24} = - 1.787$$

Then, $F' = \Phi(-1.787) = 0.037$

And finally the error bounds on F_Y at $y' = 1$,

$$F_L = \Phi\left(z' - \frac{y_U - y'}{\hat{\sigma}_Y}\right)$$

$$= \Phi\left(-1.787 - \frac{2.33 - 1}{2.24}\right)$$

$$= 0.0086$$

$$F_U = \Phi\left(z' + \frac{y' - y_L}{\hat{\sigma}_Y}\right)$$

$$= \Phi\left(-1.787 + \frac{1 - 0.273}{2.24}\right)$$

$$= 0.072$$

These bounds are plotted in Fig. 3.2.

OPTION 2 (Normal model for Y_o)

When the normal model (Option 2) for Y_o is employed, the upper and lower bounds are

$$y_L = z_L \sigma_o + z'$$

$$= (-1.64)(0.73) + 1. = -0.20$$

$$y_U = z_U \sigma_o + y'$$

$$= (1.64)(0.73) + 1. = 2.20$$

Employing the forms as above for F_L and F_U,

$$F_L = 0.10 \qquad F_U = 0.105$$

The Option 2 bounds also plotted in Fig. 3.2 do not agree well with the Option 1 bounds. Brief commentary on these differences is provided below.

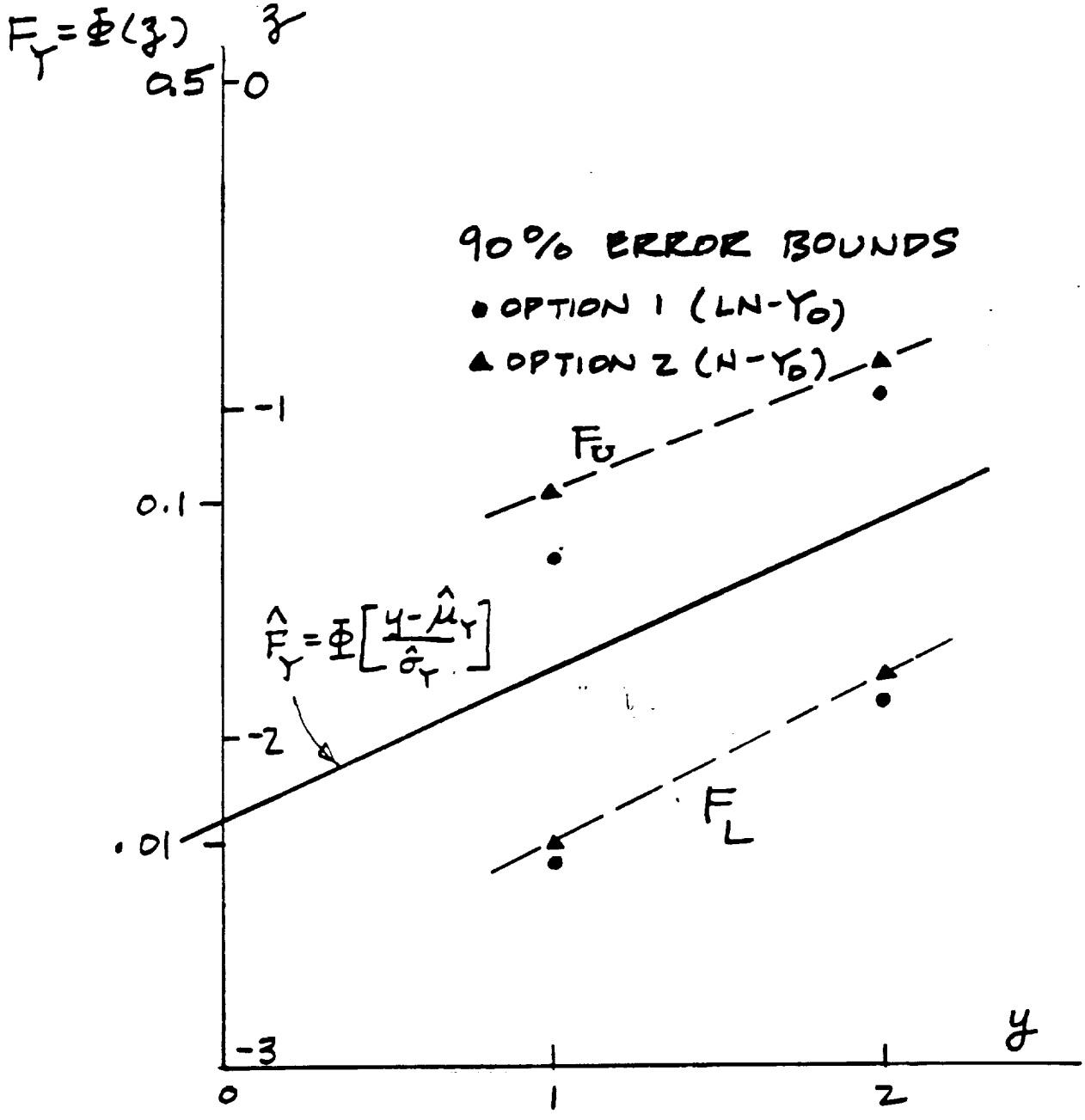


FIG 3.2 ERROR BOUNDS FOR F_Y.

Now compute the error bounds at y' = 2. Only those calculations which differ from above will be shown in the following.

The design point (See Fig. 3.1)

$r^* = -1.2$ $\qquad$ $t^* = 0.6$

$\sigma_o = 0.649$ (Eq.)

$z' = \frac{2 - 5}{2.24} = -1.34$

$F' = 0.090$

OPTION 1

$C_o = 0.324$

$\delta = 0.316$

$\tilde{Y}_o = 1.90$

$y_L = 1.11$

$y_U = 3.23$

$F_L = 0.029$

$F_U = 0.172$

OPTION 1

$y_L = (-1.64(.649) + 2. = 0.935$

$y_U = (1.64)(.649) + 2. = 3.06$

$F_L = 0.035$

$F_U = 0.193$

The error bounds are plotted on Fig. 3.2.

Some Comments

The differences in the first order error bounds reflect concerns by this author regarding the general use of the lognormal model for Y_o. As $y' \to 0$, it is noted that $C_o \to \infty$ and the model "blows up."

However, the lognormal may be a more accurate model for response variables which are guaranteed to have positive values. On the other hand, the normal Y_o model avoids any mathematical difficulties in the neighborhood of $y' = 0$. And because (a) the normal and lognormal error bounds are "reasonably close" in the region where $y' > 0$ and (b) the normal model is easier to use, it is suggested that the normal be used as a first order approximation to the error bounds.

REFERENCES

Ang, A. H. S. and Tang, W., Probability Concepts in Engineering Planning and Design, John Wiley, 1984

Thoft-Christensen, P. and Baker, M. J., Structural Reliability Theory and Applications, Springer-Verlag, 1982.

Ang, A. H. S. and Bennett, R. M., "On Reliability of Structural Systems," Proceedings of the Ship Structure Symposium, 1984.

Stuart, A. J., "Equally Correlated Variates and the Multinormal Integral," Journal of the Royal Statistical Society, Series B, Vol. 20, 1958.

Fault Tree Handbook, NUREG-0492, U. S. Nuclear Regulatory Agency, Jan. 1981.

Kjerengtroen, L., Reliability Analysis of Series Structural Systems, Ph.D. dissertation, The University of Arizona, 1984.

Kjerengtroen, L. and Wirsching, P. H., "Structural Reliability Analysis of Series Systems," Journal of Structural Engineering, Vol. 110, July 1984.

APPENDIX E

Monte Carlo Programs for Probabilistic Structural Analysis

Sueng J. Lee
Torng Yi
Paul H. Wirsching

The University of Arizona

1.0 INTRODUCTION

1.1 Introductory Remarks

Monte Carlo traditionally has been considered to be a "last resort" method for solving a probability or statistics problem because of high cost relative to accuracy of the results. However, in recent times a combination of the development of new efficient numerical techniques and new digital computing hardware have made Monte Carlo more attractive.

Presented in this report are descriptions of the following Monte Carlo programs dedicated to probabilistic structural analysis.

1. "Conventional" Monte Carlo
2. Variance reduction using antithetic variates
3. To be added later
4. To be added later

Provided in the following sections are descriptions of how each method works as well as a comprehensive study of the performance of each.

1.2 The Basic Problems

Consider the random variable Z as a function of the random vector $\underset{\sim}{X} = (X_1, X_2, \ldots X_n)$

$$Z = h(\underset{\sim}{X}) \tag{1.1}$$

The distribution of each X_i is known. It is assumed that all X_i are mutually independent.

One problem of probabilistic mechanics and design is to compute a point probability,

$$p = P[h(\underset{\sim}{X}) \leq h_o] \tag{1.2}$$

For example, p could represent the probability of exceedance of a deflection or perhaps the probability of failure.

The second problem is the extension of the first to the construction of a cumulative distribution function.

$$F_Z(z) = P[h(\underset{\sim}{X}) \leq z] \tag{1.3}$$

Clearly the two problems are identical, but optimal strategies for analysis may differ. For example, to construct the CDF, one option would be to obtain point estimates of F_Z at selected values of z, then fit a curve through the points. A second option would be to construct an empirical distribution function from a large sample of Z_i (See Sec. 2.4).

1.3 Random Samples

The basis for Monte Carlo simulation is a standard uniform distribution random number generator. Methods of generating uniform variates are generally based on recursive calculations of residues of modulus m from a linear transformation [1]. Most large computers have such a generator as a library function.

A variety of methods can be employed to generate variates from the distributions. Presented in Appendix A are algorithms used for the program presented herein.

2.0 "CONVENTIONAL" MONTE CARLO

2.1 Point Probability Estimates by Conventional Monte Carlo Using the Bernoulli Parameter

Consider a function, $h(\underset{\sim}{X})$, where $\underset{\sim}{X}$ is a vector of random variables, all having known distributions. It is required to compute,

$$p = P[h(\underset{\sim}{X}) \leq h_o] \tag{2.1}$$

The problem can be reformulated as

$$p = P[g(\underset{\sim}{X}) \leq 0] \tag{2.2}$$

where $g(\underset{\sim}{X})$, called the "performance function," is

$$g(\underset{\sim}{X}) = h(\underset{\sim}{X}) - h_o \tag{2.3}$$

In a direct Monte Carlo scheme, a sequence of K random vectors, $\underset{\sim}{X}_i$, can be sampled, and in turn, a sequence of g_i; i = 1, K computed. Define

$$Y_i = \begin{cases} 1 & \text{if } g_i \leq 0 \\ 0 & \text{if } g_i > 0 \end{cases} \tag{2.4}$$

Thus, Y_i has a Bernoulli distribution

$$P(Y_i = 1) = p \tag{2.5}$$

$$P(Y_i = 0) = 1 - p$$

where the Bernoulli parameter p is the same p as in Eq. 2.1.

The maximum likelihood estimate (MLE) of p is [5],

$$\hat{p} = \frac{1}{K} \sum_{i=1}^{K} Y_i \tag{2.6}$$

But $\sum Y_i$ is just the total number of $g_i \leq 0$, denoted as N_o. Thus, $\hat{p}$ is just the fraction of the g_i's less than zero

$$\hat{p} = \frac{N_o}{N} \tag{2.7}$$

A flow diagram of conventional Monte Carlo is given in Fig. 2.1. A listing of a computer program for conventional Monte Carlo employing the Bernoulli parameter is provided in Appendix B and an example of the output is shown in Fig. 2.2.

2.2 Confidence Intervals on the Bernoulli Parameter, p

The MLE of p is $\hat{p}$. Because of sampling error, $\hat{p}$ is only an estimate, and the key question is how close is p to $\hat{p}$. Confidence intervals are described below. Note that these confidence intervals refer to sampling error of the Monte Carlo process, not uncertainties associated with the parameters of X_i.

Consider $\hat{p}$,

$$\hat{p} = \frac{1}{K} \sum_{i=1}^{K} Y_i \tag{2.8}$$

The mean and variance of $\hat{p}$ are [5]

$$E(\hat{p}) = p \tag{2.9}$$

$$V(\hat{p}) = \frac{p(1 - p)}{K} \tag{2.10}$$

By the central limit theorem, $\hat{p}$ will approach a normal distribution as $K \to \infty$. Confidence intervals for p are constructed using normal distribution mathematics,

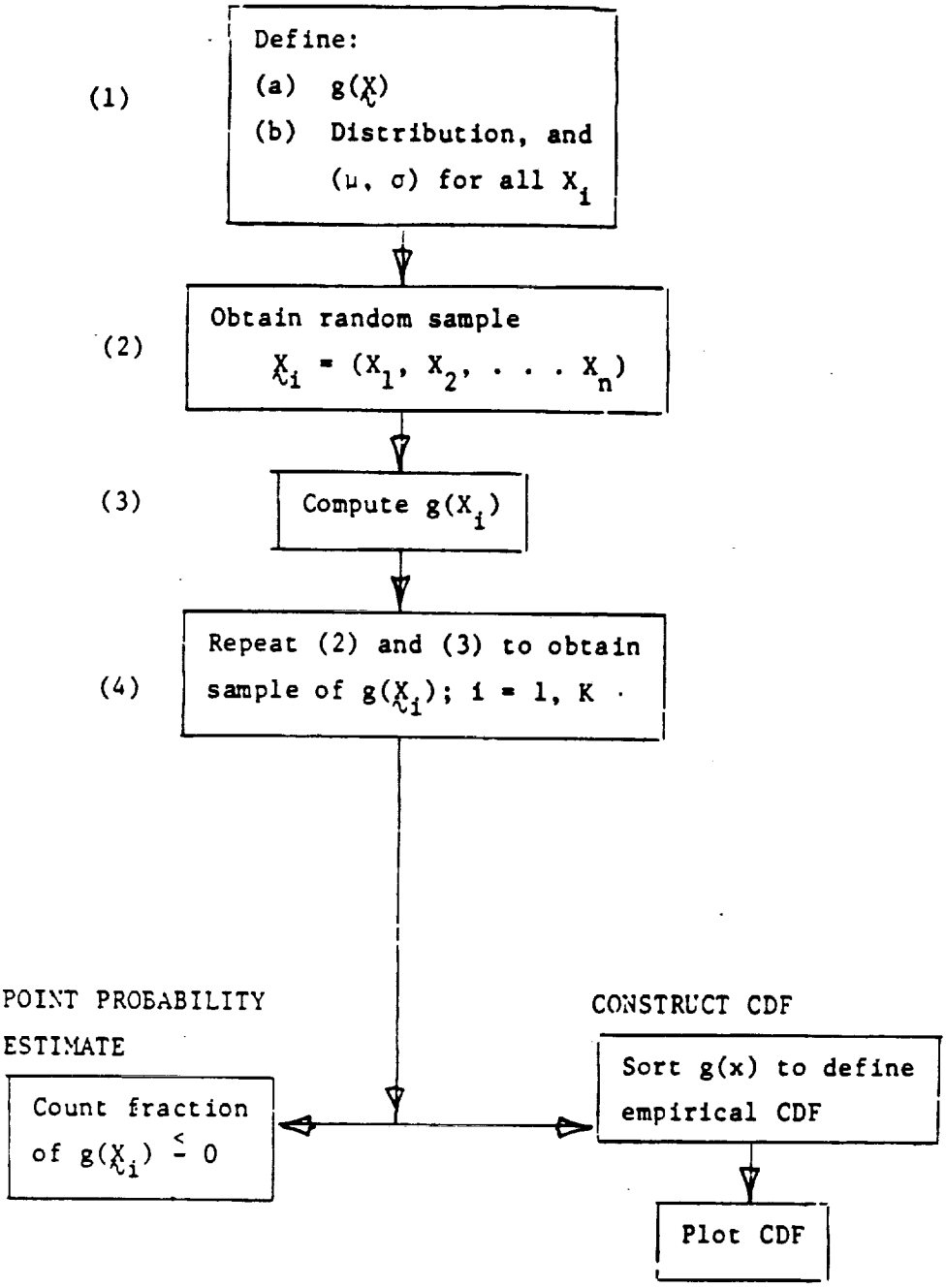


Fig. 2.1 Flow diagram of conventional Monte Carlo

MONTE CARLO SOLUTION

LIMIT STATE FUNCTION : R=S

SAMPLE SIZE, K= 100

NUMBER OF RANDOM VARIABLES, N= 2

RANDOM VARIABLES

VARIABLE	DISTRIBUTION	MEAN	STD DEV
R	WEIBULL	.20000E+02	.20000E+01
S	EVD	.10000E+02	.20000E+01

STATISTICS OF Y :

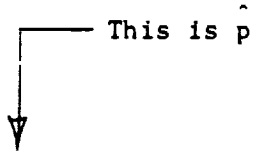
Note that Y is the same as g(X); these are the statistics on the limit state function.

MEAN = .10018E+02

STD DEV = .27499E+01

MEDIAN = .96606E+01

COV = .27450E+00



This is $\hat{p}$

NUMBER OF NEG Y VALUES= 0. PERCENT OF TRIALS= .000000

Fig. 2.2 Output of conventional Monte Carlo program. (No sorting requested)

Performance function; g(R,S) = R - S

$$\hat{p} - z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{K}} \leq p \leq \hat{p} + z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{K}} \qquad (2.11)$$

where $\hat{p}$ is substituted for p in the variance. The probability that p will be bounded by the lower and uppper limit is $1-\alpha$, where α is the confidence coefficient. $z_{\alpha/2}$ is the standard normal variate corresponding to $\alpha/2$. Commonly used values

α	$z_{\alpha/2}$
.10	1.64
.05	1.96
.01	2.58

The confidence interval of Eq. 2.11 relies on the central limit theorem and must be considered as only an approximation for finite K. In general, the approximation is considered "valid" if $Kp > 5$ [5].

Eq. 2.11 can be written as,

$$\hat{p}(1-\gamma) \leq p \leq \hat{p}(1+\gamma) \qquad (2.12)$$

where,

$$\gamma = \frac{z_{\alpha/2}}{\hat{p}}\sqrt{\frac{\hat{p}(1-\hat{p})}{K}} \qquad (2.13)$$

Eq. 2.13 is displayed in Figs. 2.3 and 2.4 for 90% and 95% confidence intervals respectively. These figures show the sample size requirements for confidence intervals of a given width and level. For example, if the point probability is expected to be about 10^{-3}, and it is required to have p within $\pm$ 10% of p with a confidence of 90%, then it is necessary to have a sample of size $K > 200{,}000$.

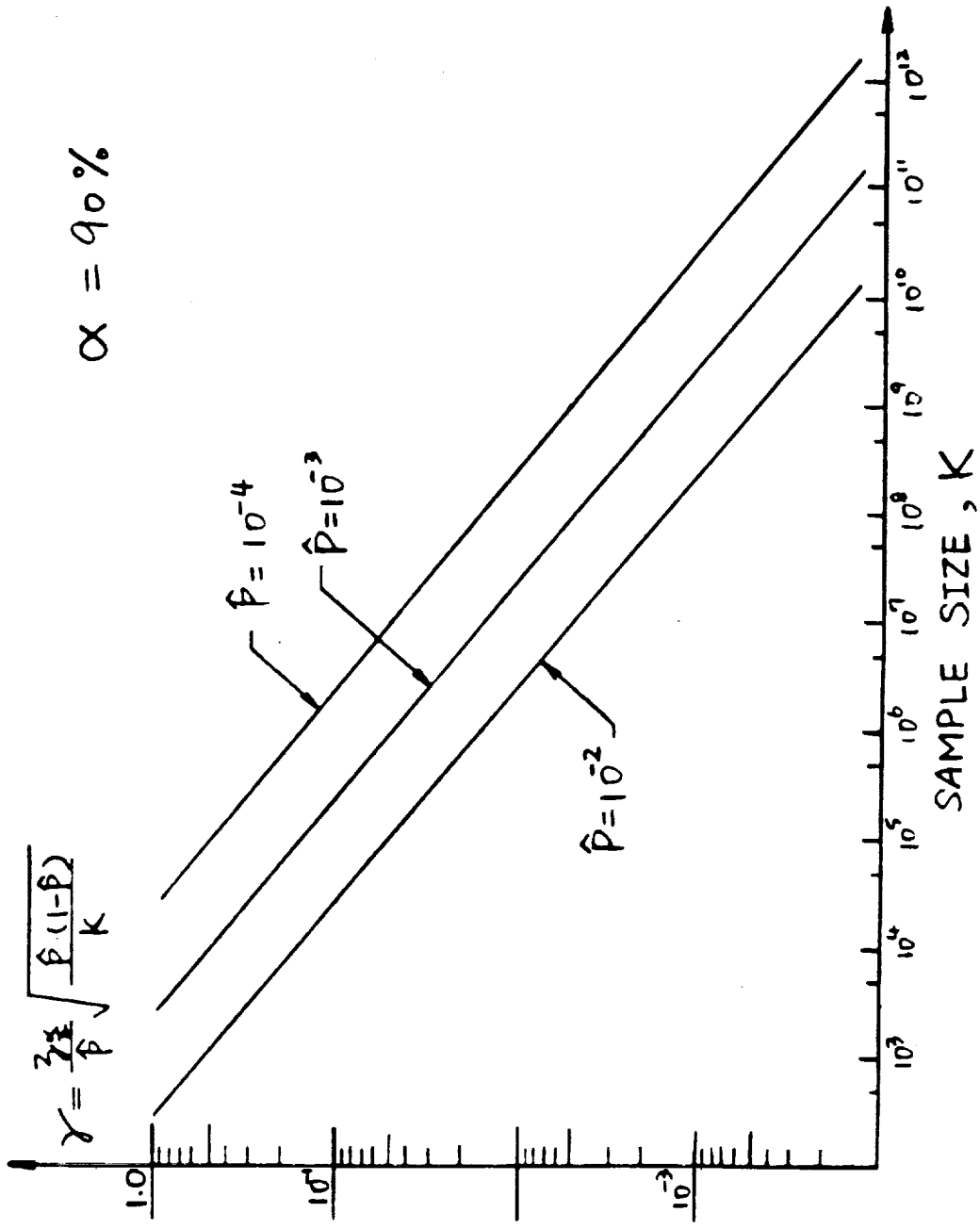


Fig. 2.3 90% confidence intervals on p as a function of sample size and $\hat{p}$

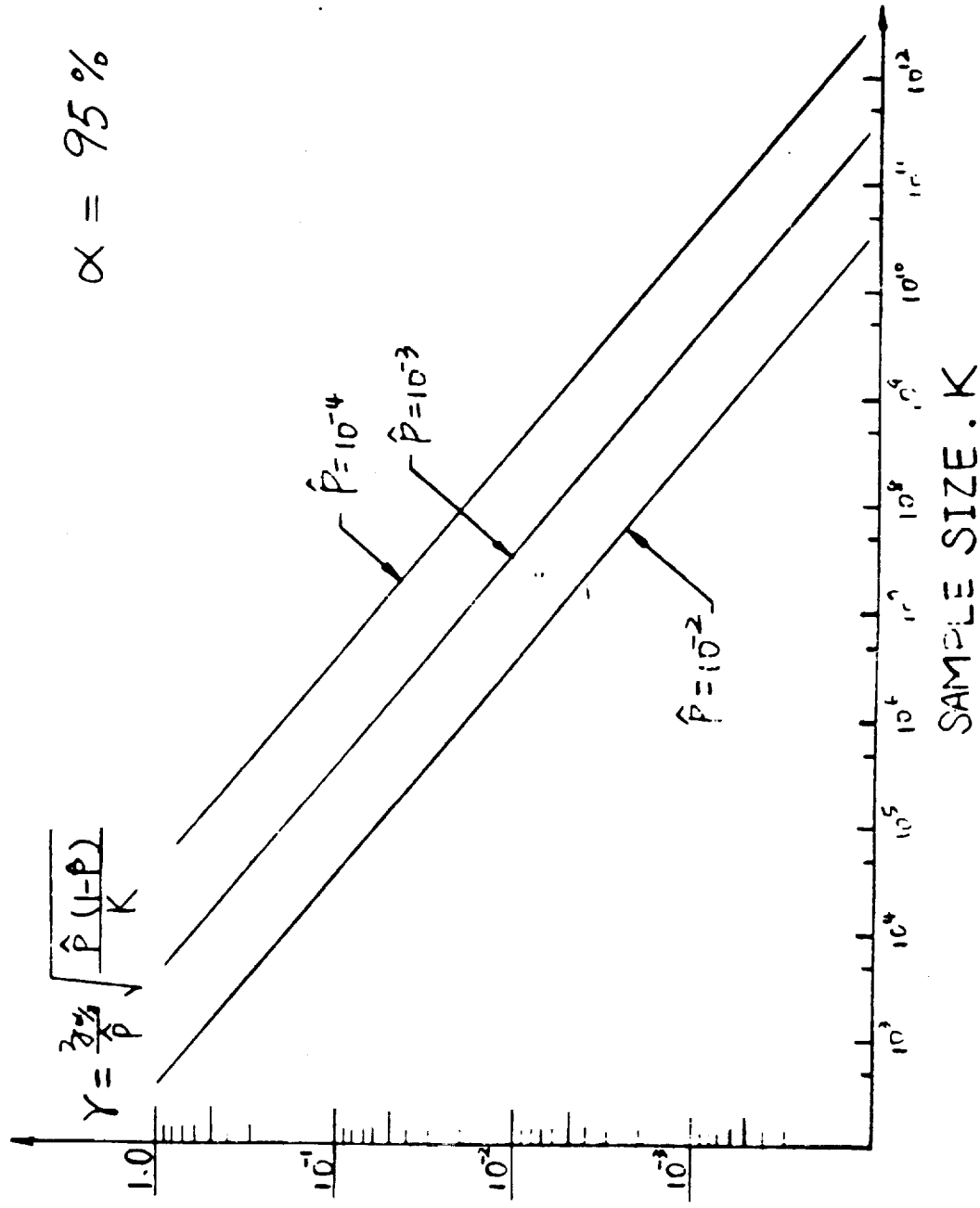


Fig. 2.4 95% confidence intervals on p as a function of sample size and $\hat{p}$.

2.3 Computer CPU Time on the CYBER 175

The conventional Monte Carlo program of Appendix B was exercised on several problems using all five of the available distributions. CPU time was recorded for each program. It is assumed that this conventional Monte Carlo program will provide an upper bound to CPU time relative to other, and more efficient, Monte Carlo schemes. The CYBER 175 is the mainframe computer at the University of Arizona, and all results relate to this machine.

Recorded CPU time for several examples was consistent. Compilation and loading time for all cases are shown in Table 2.1. These are average values, but there was little variation.

Execution CPU time essentially depends only upon the number of variables and not on distributional forms or performance functions. Fig. 2.5 illustrates the CPU execution time per variate as a function of sample size K. Total CPU time is obtained by adding compilation and loading time to execution time.

A sample program was run on both the CYBER 175 and the VAX 11/780 for a time comparison. The results shown in Table 2.2, reaffirm the fact that the VAX is too slow for production Monte Carlo.

To get an idea of computer charges for running Monte Carlo, Fig. 2.6 is provided. This is the commercial rate of the UA CYBER 175 for low priority jobs.

Table 2.1

Compilation and Loading CPU Time for Conventional Monte Carlo on CYBER 175 Program

	CPU Time (sec)
Compile	1.0
Load	0.25

Table 2.2

Comparison of CPU time Between CYBER 175 and VAX 11/780 for one Example Problem*

	Time (sec)	
	CYBER 175	VAX 11/780
Compile	1.0	14
Link	0.25	5
Execution	7.5	30
TOTAL	8.75	49.0

*There were 2 variables; K = 30,000.

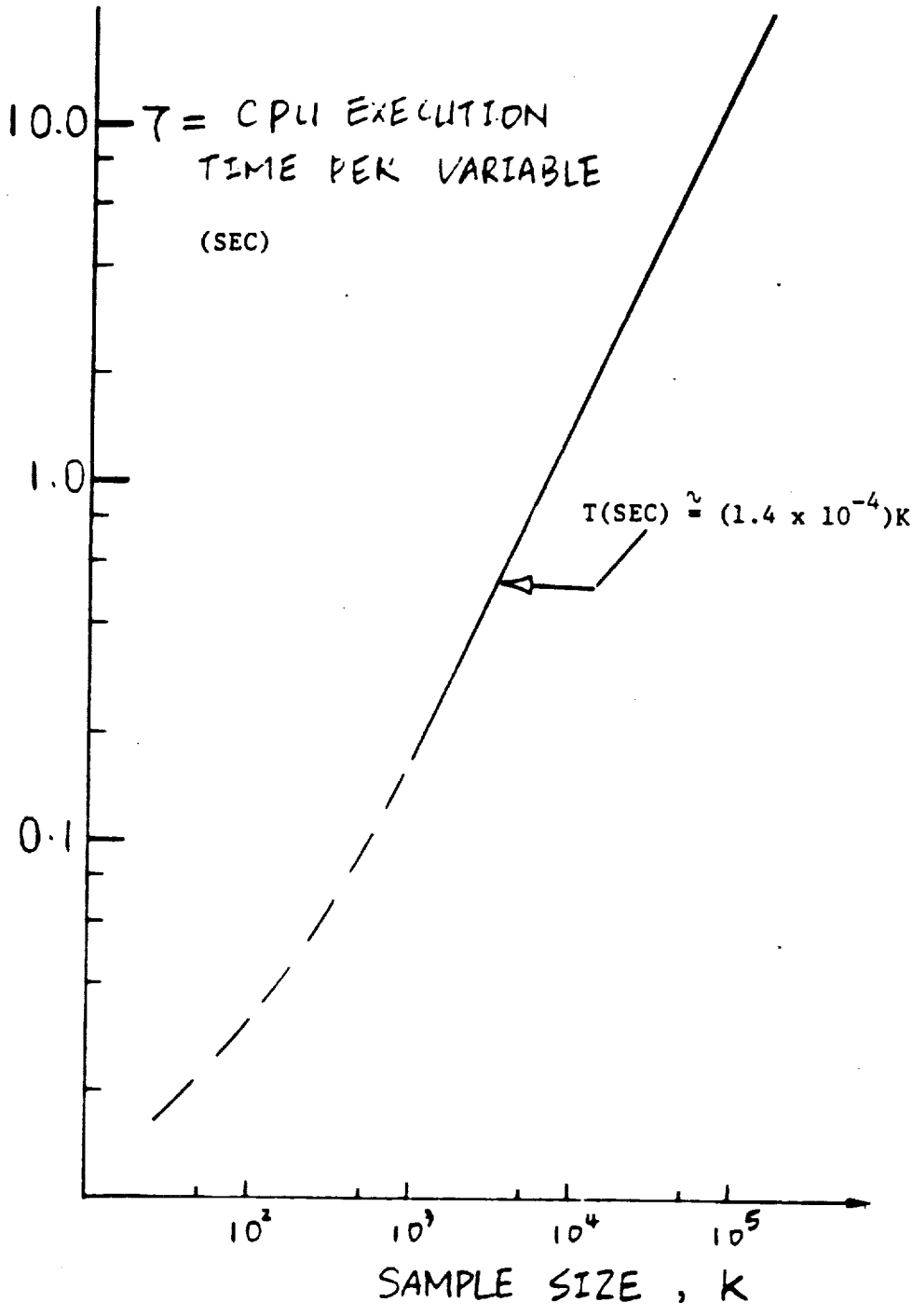


Fig. 2.5 CPU execution time per variate on CYBER 175 as a function of sample size K.

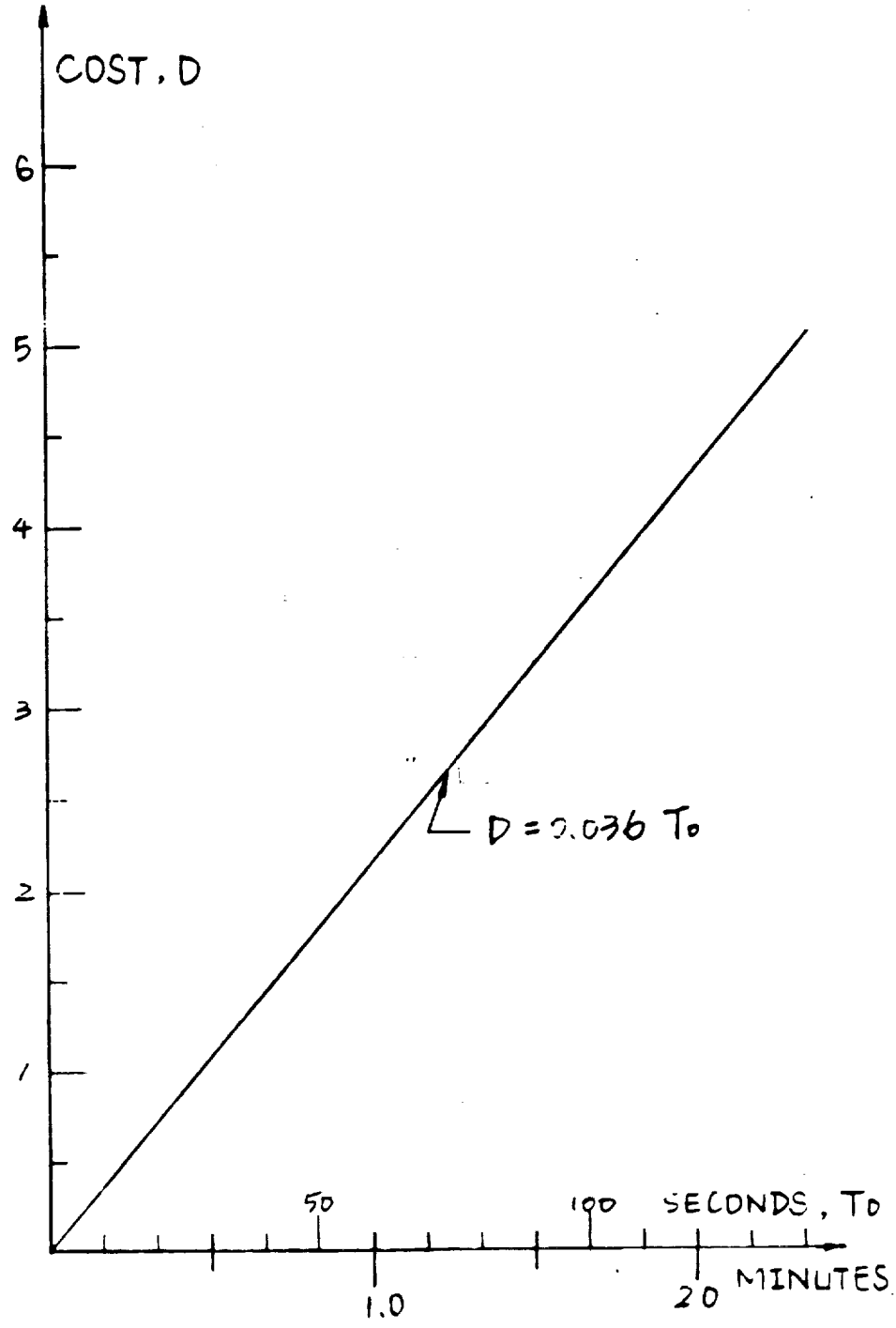


Fig. 2.6 Cost in dollars ($), D, as a function of time for the UA CYBER 175; lowest priority.

2.4 Comparison of Monte Carlo to Wu/FPI

Computational efficiency was the motivation for the development of the Wu/FPI program. It is generally known that Monte Carlo is inefficient relative to a fast probability integration method. An attempt is made here to quantify differences in computer time between conventional Monte Carlo and Wu/FPI. Because the cost of conventional Monte Carlo depends upon the accuracy and probability level required, a general direct comparison can't be made. However, an example presented in the following clearly demonstrates the high cost of Monte Carlo.

Suppose that it is required to provide a Monte Carlo solution such that the 90% CI for p is within $\pm$ 10% of $\hat{p}$. The CPU execution time for the CYBER 175 can be computed from Figs. 2.3 and 2.5 for a given probability level, $\hat{p}$. This CPU time is shown in Fig. 2.7 as a function of the number of variables in $g(\underset{\sim}{X})$ for $\hat{p} = 10^{-3}$ and 10^{-4}. At these levels Monte Carlo is two to three orders of magnitude more expensive than FPI. And the FPI sclution is likely to be more accurate. Moreover, for smaller tail probabilities FPI gets no more expensive while Monte Carlo will break the bank.

2.5 Estimating the CDF of a Random Function

2.5.1 The Empirical CDF

Conventional Monte Carlo provides capability for estimating the complete distribution function of a function of random variables. Define the random variable Z, as a function of the random vector $\underset{\sim}{X}$.

$$Z = Z(\underset{\sim}{X}) \tag{2.14}$$

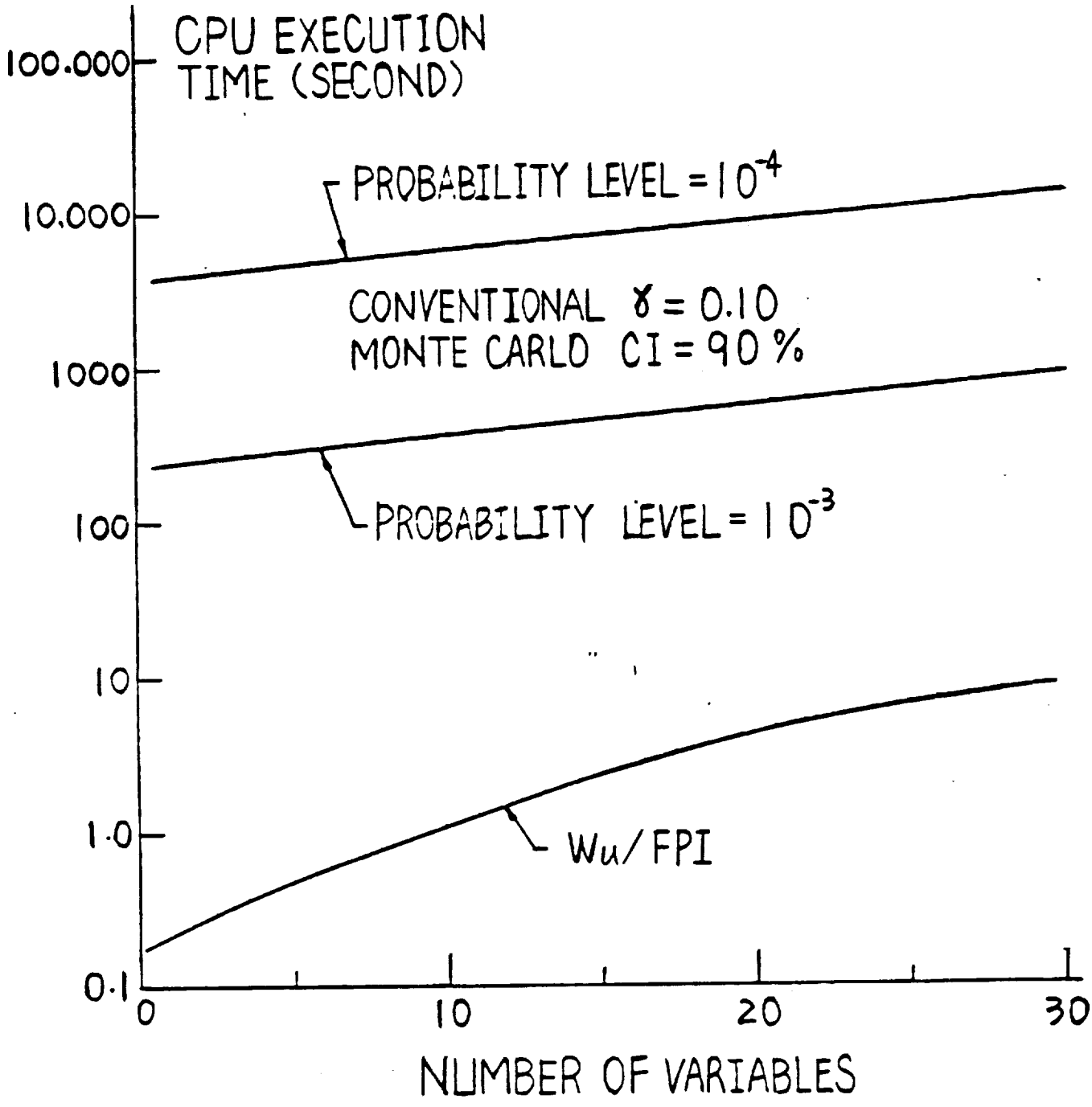


Fig. 2.7 Comparison of Monte Carlo and Wu/FPI CPU time

A random sample of X_i; i = 1, K is used to generate a random sample of Z_i; i = 1, K. In turn, an empirical distribution function of Z can be constructed using methods of probability plotting. The empirical CDF, denoted as F_i, will be an estimate of the CDF of Z, $F_Z(z)$.

Various forms of F_i have been proposed [3, 4, 6]. The values of F_i below correspond to $Z_{(i)}$ where $Z_{(i)}$ is the ith smallest value of the random vector $\underset{\sim}{Z}$. Thus, $F_i \equiv F_i(Z_{(i)})$.

1. Hazen; $F_i = \frac{i - 1/2}{K}$
2. Gumbel; $F_i = \frac{i}{K + 1}$
3. Median ranks, $F_i = \frac{i - 0.3}{n + 0.4}$

Through prior experience on extensive Monte Carlo simulation, this author has found that the Hazen formula consistently provides "good estimates" of F_Z.

2.5.2 The Sort Routine

To construct the empirical CDF it is required to sort the random sample $\underset{\sim}{Z}$ to obtain an ordered sample $\underset{\sim}{Z}_o$. Let $Z_{(i)}$ denote the ith smallest value.

The routine used in this Monte Carlo code is program QUICKSORT which is considered to be the fastest available [7]. A description of QUICKSORT is given in Appendix C. The Fortran statements for this code are provided in the program listing in Appendix B.

CPU time requirements for the sort routine can be relatively large for large samples. Fig. 2.8 shown CPU execution times as a function of the size of the $\underset{\sim}{Z}$ vector.

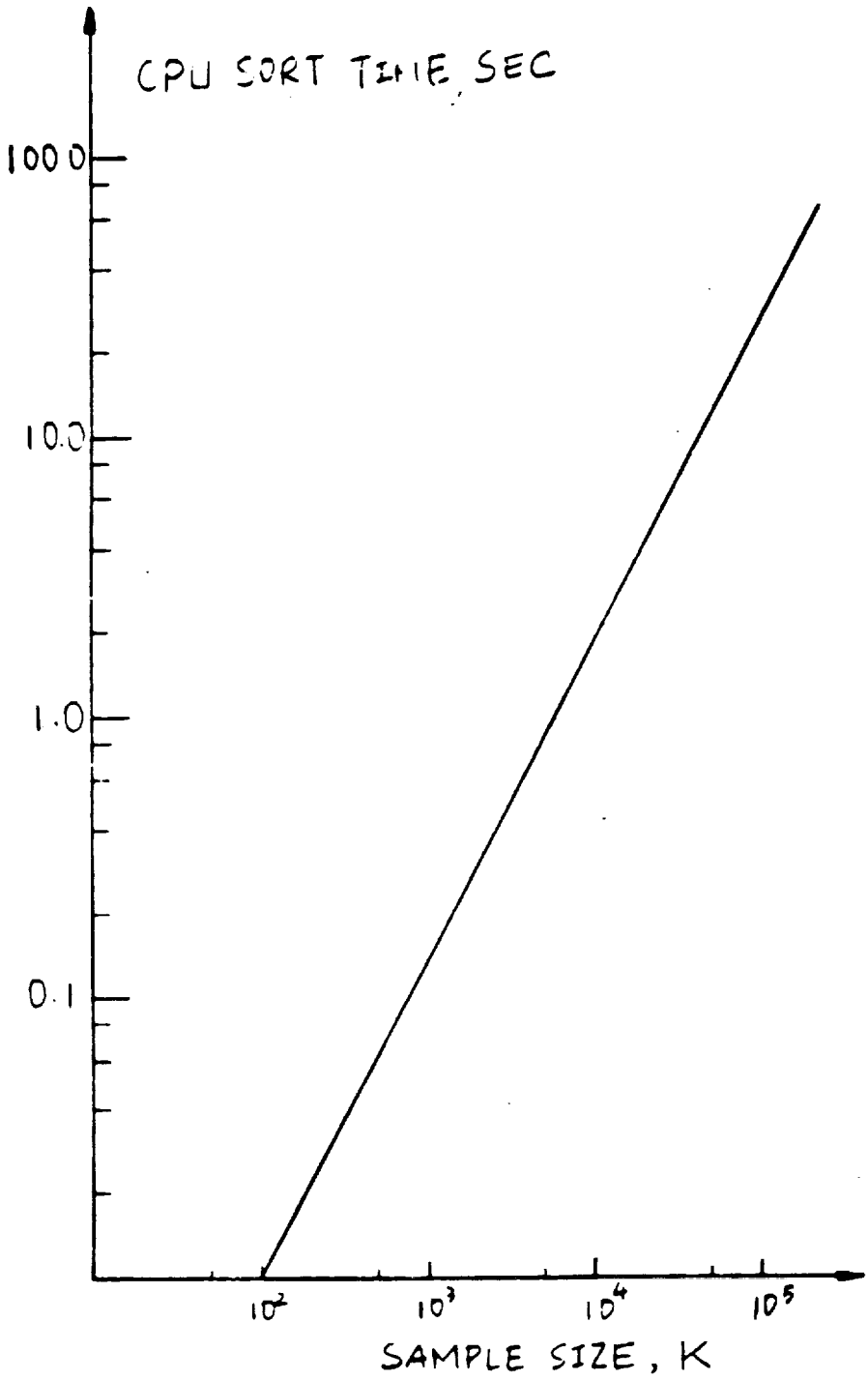


Fig. 2.8 CPU sort time (execution) as a function of sample size for the CYBER 175.

2.5.3 An Example.

Shown in Fig. 2.9 is a table of the sorted vector $Z_{(i)}$ and the corresponding F_i for the example of Fig. 2.1. This is the data required for plotting. The empirical CDF of Fig. 2.10 was done by hand, but in general such graphs can be automated using a computer graphics package.

SORTED VALUES OF Z AND THE EMPIRICAL CDF

I =	1	.32159E+01	.40876E+01	.42831E+01	.44764E+01	.45626E+01
I =	6	.48457E+01	.48984E+01	.50586E+01	.56150E+01	.59102E+01
I =	11	.59944E+01	.60426E+01	.66202E+01	.68500E+01	.69210E+01
I =	16	.69827E+01	.70597E+01	.70685E+01	.70780E+01	.71004E+01
I =	21	.76156E+01	.79653E+01	.83861E+01	.84534E+01	.84720E+01
I =	26	.87304E+01	.87709E+01	.87964E+01	.88850E+01	.89137E+01
I =	31	.90619E+01	.90971E+01	.91454E+01	.92372E+01	.92557E+01
I =	36	.92816E+01	.92823E+01	.93259E+01	.95770E+01	.95829E+01
I =	41	.95862E+01	.95993E+01	.96380E+01	.98157E+01	.98782E+01
I =	46	.10054E+02	.10115E+02	.10137E+02	.10256E+02	.10370E+02
I =	51	.10376E+02	.10581E+02	.10607E+02	.10631E+02	.10644E+02
I =	56	.10712E+02	.10771E+02	.10773E+02	.10791E+02	.10846E+02
I =	61	.10856E+02	.10874E+02	.10958E+02	.11125E+02	.11162E+02
I =	66	.11191E+02	.11246E+02	.11344E+02	.11409E+02	.11616E+02
I =	71	.11730E+02	.11760E+02	.11802E+02	.11912E+02	.11933E+02
I =	76	.12122E+02	.12140E+02	.12284E+02	.12413E+02	.12573E+02
I =	81	.12667E+02	.12803E+02	.12844E+02	.12867E+02	.12873E+02
I =	86	.12893E+02	.12963E+02	.13042E+02	.13131E+02	.13142E+02
I =	91	.13273E+02	.13297E+02	.13361E+02	.13638E+02	.13709E+02
I =	96	.13943E+02	.14797E+02	.14983E+02	.15123E+02	.15305E+02

I =	1	.50000E-02	.15000E-01	.25000E-01	.35000E-01	.45000E-01
I =	6	.55000E-01	.65000E-01	.75000E-01	.85000E-01	.95000E-01
I =	11	.10500E+00	.11500E+00	.12500E+00	.13500E+00	.14500E+00
I =	16	.15500E+00	.16500E+00	.17500E+00	.18500E+00	.19500E+00
I =	21	.20500E+00	.21500E+00	.22500E+00	.23500E+00	.24500E+00
I =	26	.25500E+00	.26500E+00	.27500E+00	.28500E+00	.29500E+00
I =	31	.30500E+00	.31500E+00	.32500E+00	.33500E+00	.34500E+00
I =	36	.35500E+00	.36500E+00	.37500E+00	.38500E+00	.39500E+00
I =	41	.40500E+00	.41500E+00	.42500E+00	.43500E+00	.44500E+00
I =	46	.45500E+00	.46500E+00	.47500E+00	.48500E+00	.49500E+00
I =	51	.50500E+00	.51500E+00	.52500E+00	.53500E+00	.54500E+00
I =	56	.55500E+00	.56500E+00	.57500E+00	.58500E+00	.59500E+00
I =	61	.60500E+00	.61500E+00	.62500E+00	.63500E+00	.64500E+00
I =	66	.65500E+00	.66500E+00	.67500E+00	.68500E+00	.69500E+00
I =	71	.70500E+00	.71500E+00	.72500E+00	.73500E+00	.74500E+00
I =	76	.75500E+00	.76500E+00	.77500E+00	.78500E+00	.79500E+00
I =	81	.80500E+00	.81500E+00	.82500E+00	.83500E+00	.84500E+00
I =	86	.85500E+00	.86500E+00	.87500E+00	.88500E+00	.89500E+00
I =	91	.90500E+00	.91500E+00	.92500E+00	.93500E+00	.94500E+00
I =	96	.95500E+00	.96500E+00	.97500E+00	.98500E+00	.99500E+00

Fig. 2.9 Sorted Z_i and corresponding empirical CDF for the example of Fig. 2.1

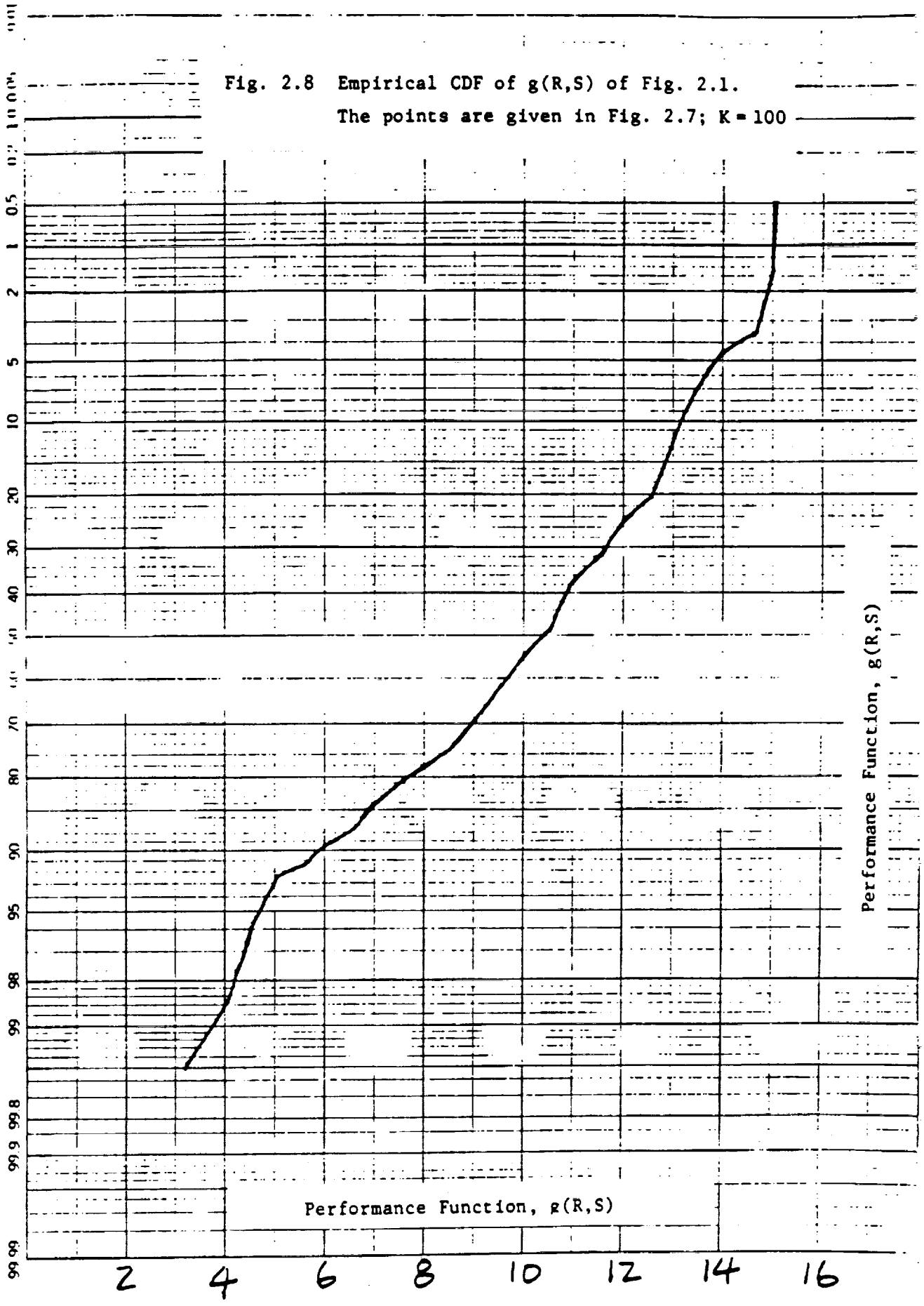


Fig. 2.8 Empirical CDF of g(R,S) of Fig. 2.1. The points are given in Fig. 2.7; K = 100

3.0 THE VARIANCE REDUCTION METHOD

3.1 Preliminary Remarks

The variance of Monte Carlo estimators can be reduced, relative to straightforward sampling of Chapt. 2.0, by appropriate operations with negatively correlated samples. Ang and Tang [1] present several examples which demonstrate dramatic improvements in efficiency realized by variance reduction methods.

A variance reduction computer program, tailored for structural mechanics analysis by providing point probability estimates of functions of random variables has been developed. The listing is given in Appendix D. To assess performance, the program has been exercised on several examples. Results presented in Section 3.6 show dramatic improvement of variance reduction over conventional Monte Carlo in some cases. In other cases, the improvement is only modest. Some general conclusions are presented in Section 3.7. For the most part however, for a given problem it is difficult to predict how much improvement one can expect with variance reduction.

3.2 The Essence of Variance Reduction

The goal of analysis is to estimate

$$p = P[h(\underset{\sim}{X}) < h_o] \tag{3.1}$$

Suppose $\hat{p}$ and $\hat{p}'$ are two unbiased estimates of p. (The method for obtaining a point estimate of p is described in Sec. 3.4 below.) The two estimators may be combined to form another estimator

$$\bar{p} = \frac{1}{2}(\hat{p} + \hat{p}') \tag{3.2}$$

The expected value of p_E is,

$$E(\bar{p}) = \frac{1}{2}[E(\hat{p}) + E(\hat{p}')] = p \tag{3.3}$$

which means that $\bar{p}$ is an unbiased estimator.

The corresponding variance is

$$V(\bar{p}) = \frac{1}{4}[V(\hat{p}) + V(\hat{p}') + 2\ \mathrm{Cov}\ (\hat{p}, \hat{p}')] \tag{3.4}$$

If p and p' are statistically independent, for example, based on two separate and independent sets of random numbers,

$$V(\bar{p}) = \frac{1}{4}\ [V(\hat{p}) + V(\hat{p}')] \tag{3.6}$$

Thus, the accuracy of the estimator $\bar{p}$ can be improved over that of the independent case if $\hat{p}$ and $\hat{p}'$ are <u>negatively</u> correlated. Ang and Tang cite several examples (no structural analysis) where variance reduction can provide a dramatic improvement in efficiency of probability estimation [1].

An estimate of p is obtained by several samples, $\bar{p}_i$; i = 1,K.

$$p_E = \frac{1}{K} \sum_{i=1}^{K} \bar{p}_i \tag{3.7}$$

all $\bar{p}_i$ are independent. Note that p_E will approach normality as $K \to \infty$ as a consequence of the central limit theorem.

The mean and variance of p_E are,

$$E(p_E) = p \tag{3.8}$$

$$V(p_E) = \sigma_p^2/K \tag{3.9}$$

where σ_p^2 is estimated as,

$$s_p^2 = \frac{1}{K-1} \sum_{i=1}^{K} (\bar{p}_i - p_E)^2 \tag{3.10}$$

3.3 How to Obtain Negatively Correlated Samples

Suppose that the uniformly distributed variate u_i is used to generate a number x_i from a given distribution (See Appendix A). Then the uniform variate $u_i' = 1 - u_i$ will produce an x_i' such that x_i and x_i' will be negatively correlated. The u_i' are called "antithetic" variates.

And in general, if $u_1, u_2, \ldots u_n$ is used to generate $\hat{p}$, and $1 - u_1$, $1 - u_2, \ldots 1 - u_n$ is used to generate $\hat{p}'$, then $\hat{p}$ and $\hat{p}'$ will be negatively correlated.

Such a procedure works well when the integral transform is used, e.g., Weibull, EVD. One uniform variate u_i is used to generate one x_i. But where Box-Muller is used to generate normal variates, two u_i are chosen (See Appendix A). While the resulting x_i and x_i' will be negatively correlated, the correlation coefficient will not be -1.0. An improvement can be made by choosing x_i' as a "mirror image" of x_i in the distributions. This can be done by

$$x_i' = 2\mu - x_i \tag{3.11}$$

where μ is the mean of X.

3.4 How to Obtain Point Probability Estimates

3.4.1 The Two Variable Case

The structural reliability problem in which p is the probability of failure will be used to illustrate how $\hat{p}$ and $\hat{p}'$ are obtained. Consider the design case where the two variables are R (strength) and S (stress). Estimate p, where

$$p = P[R - S \leq 0] \tag{3.12}$$

Both R and S are random variables whose density functions are shown in Fig. 3.1. First S, having been identified as the variable having the largest variance, is the "reference." A random variate R_i is sampled from the other factor, R. An estimate of p is

$$\hat{p}_i = P(S > R_i)$$

$$= 1 - F_S(R_i) \tag{3.13}$$

where F_S is the CDF of S.

It should now be apparent why sampling is done on the smallest variance term. $\hat{p}$ is a "good" estimate of p if the distribution is narrow, and is exact as $\sigma_R \to 0$.

Now the antithetic variate R_i' is sampled as described above. Because it is negatively correlated to R_i, its position relative to R_i will be as shown in Fig. 3.2. Then,

$$\hat{p}_i' = P(S > R_i')$$

$$= 1 - F_S(R_i') \tag{3.14}$$

and the ith estimate of p is

$$\bar{p}_i = \frac{1}{2}(\hat{p}_i + \hat{p}_i') \tag{3.15}$$

As a second example, consider again the case where R and S are the basic variables, but now where $\sigma_R < \sigma_S$. In this case, R would be the reference variable. Random points S_i and the antithetic variate S_i' are sampled from S. The estimates now are,

$$\hat{p}_i = F_R(S_i) \tag{3.16}$$

$$\hat{p}_i' = F_R(S_i')$$

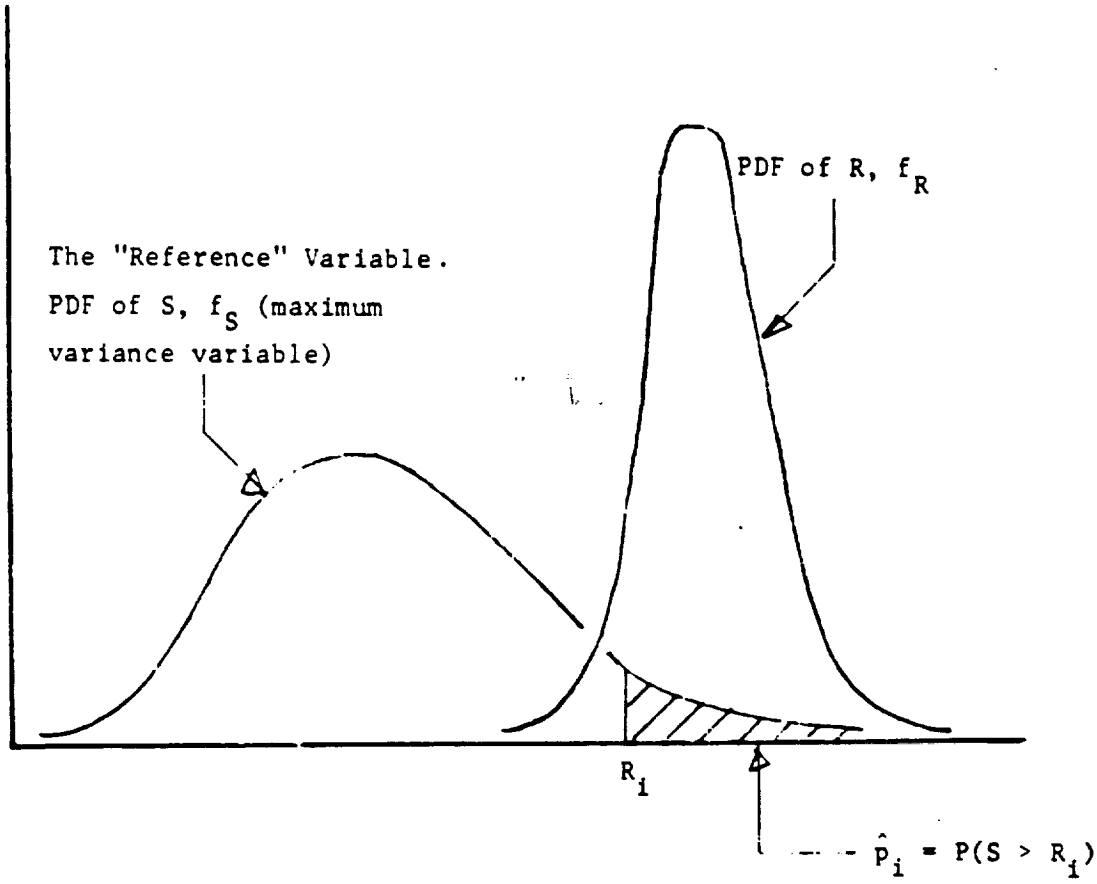


Fig. 3.1 Estimate of p using one point sampled from the minimum variance variable.

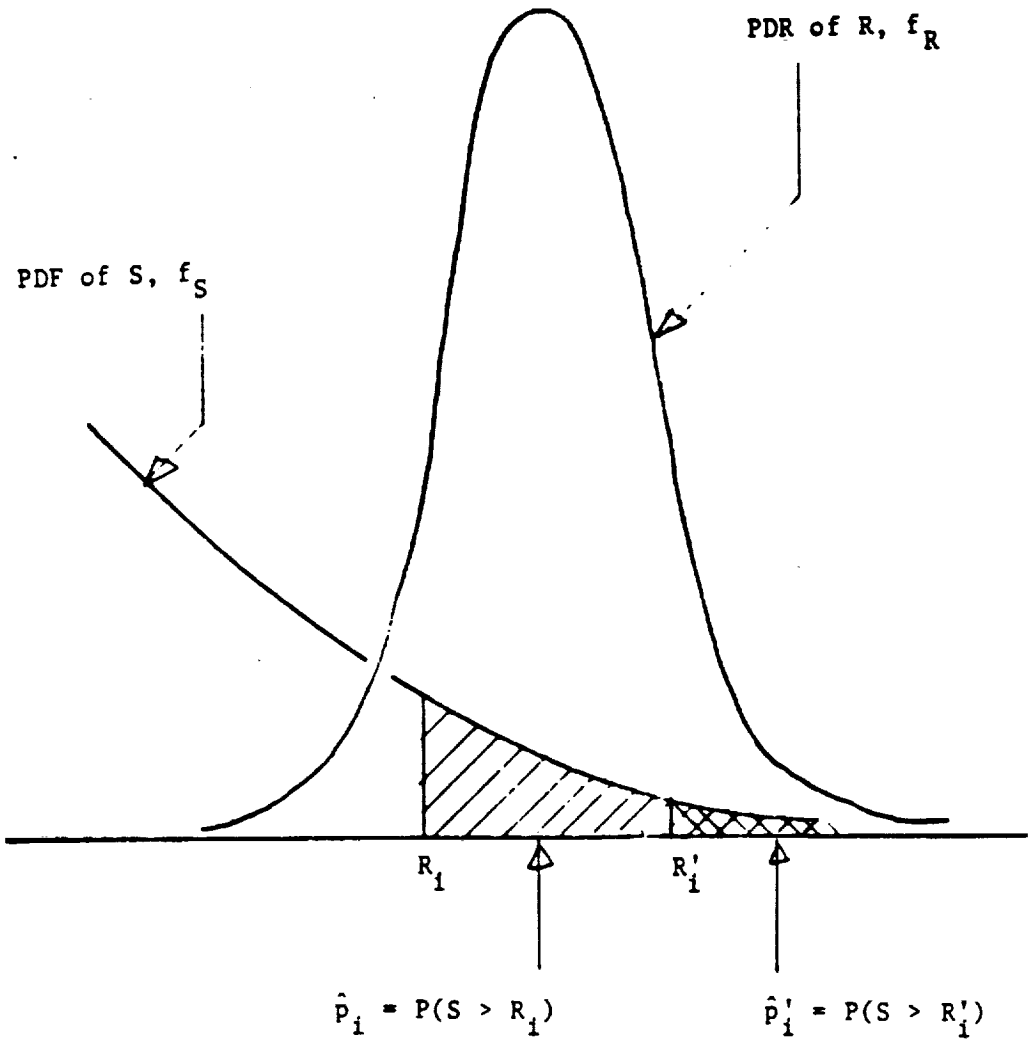


Fig. 3.2 Estimates of p using a point R_i sampled from R and the antithetic variate of R_i, denoted as R_i'

Thus, it is seen that the variable type (stress or strength) must be identified to obtain the proper form for computing estimates.

Fig. 3.2 shows why negatively correlated variables tend to provide good estimates. Being on both sides of a distribution, R_i and R_i' combine to produce an "average" estimate of p.

3.4.2 The General Case

In general, the performance function, $g(\underset{\sim}{X}) = h(\underset{\sim}{X}) - h_o$ is a non-linear function of several variables. The method of obtaining a point estimate of p is an extension of the scheme for two variables.

The reference variable is defined, not as the one having the maximum variance, but rather the one having the maximum impact. For example, if

$$g = 5R - S \tag{3.17}$$

and $\sigma_R = \sigma_S/2$, clearly the random variable, $R_1 = 5R$ will have a larger variance than S. Thus, we say that R is the maximum impact variable.

In general, the maximum impact variable can be found by estimating $\partial g/\partial X_i$ for each X_i. The maximum impact variable, denoted as X_M, is that X_i for which $|\partial g/\partial X_i|$ is the largest.

The sign of $\partial g/\partial X_i$ identifies variable type; <u>stress</u> if (+) and strength if (-). As indicated above, the "type" of X_M must be known to choose the appropriate form for estimating p (e.g., Eqs. 3.13 and 3.16).

The estimates $\hat{p}$ and $\hat{p}'$ proceed as follows. Sample all variables but X_M. Let $g(\underset{\sim}{X}) = 0$, and solve for x_M (this is done by the secant method in the program).

$$x_M = h(\underset{\sim}{x}_o) \tag{3.18}$$

where $\underset{\sim}{x}_o$ is the vector of sampled $\underset{\sim}{X}$ minus X_M.

The estimate of p is,

$$\hat{p} = \begin{cases} F_{X_M}(x_M) \text{ if } X_M \text{ is a strength variable} \\ 1 - F_{X_M}(x_M) \text{ if } X_M \text{ is a stress variable} \end{cases} \quad (3.19)$$

To obtain $\hat{p}'$, the antithetic vector $\underline{x}_o'$ of $\underline{x}_o$ is used in Eq. 3.19.

3.5 Confidence Intervals on p

Noting that p_E is normally distributed, approximate $1 - \alpha$ confidence intervals on p can be constructed as [5],

$$p_E - \frac{z_{\alpha/2}\, s_p}{\sqrt{K}} < p < p_E + \frac{z_{\alpha/2}\, s_p}{\sqrt{K}} \quad (3.20)$$

or,

$$p_E(1 - \gamma) < p < p_E(1 + \gamma) \quad (3.21)$$

where,

$z_{\alpha/2}$ = standard normal variate (absolute value) at probability level $\alpha/2$.

$$\gamma = \frac{z_{\alpha/2}\, C_p}{\sqrt{K}} \quad (3.22)$$

$$C_p = s_p / p_E \quad (3.23)$$

The UA variance reduction program chooses K to produce a specific confidence interval. For example, if you want to sample until the 95% confidence intervals are $\pm$ 10% of p_E,

$$\gamma = 0.10 \qquad z_{\alpha/2} = 1.64 \quad (3.24)$$

and solving Eq. 3.22 for K,

$$K > \frac{z_{\alpha/2}^2 C_p^2}{\gamma} = 269\ C_p^2 \qquad (3.25)$$

To find C_p, an initial sample of K = 1000 is chosen and an estimate of C_p is obtained. Then if K < 1000 in Eq. 3.16, the process is terminated with narrower confidence intervals than requested. If K > 1000, the program will continue to sample to that value.

3.6 The Variance Reduction Monte Carlo Program

A flow diagram which outlines the logic of the variance reduction program is provided in Fig. 3.3. Sample output of the program is shown in Fig. 3.4 with some commentary.

Two versions of the program have been developed. An interactive version (IVARED) runs on the IBM PC/XT. Program VARED runs on the VAX or CYBER 175. A listing of VARED is given in Appendix D.

3.7 Examples of the Performance of VARED

Twelve examples of the use of VARED to produce point probability estimate are provided in Tables 3.1 through 3.12. Point estimates by VARED are compared to the exact solution (closed form or POFAIL) if available. The exact solution, provided by program POFAIL, is employed for performance functions involving two variables. For larger problems, Wu/FPI is used. For the VARED solutions, 95% confidence intervals (α = 5%) are specified along with γ = 0.10.

To compare variance reduction with conventional Monte Carlo, sample size requirements and CPU time for the latter are extracted from Figs. 2.4 and 2.5 and are presented in the tables.

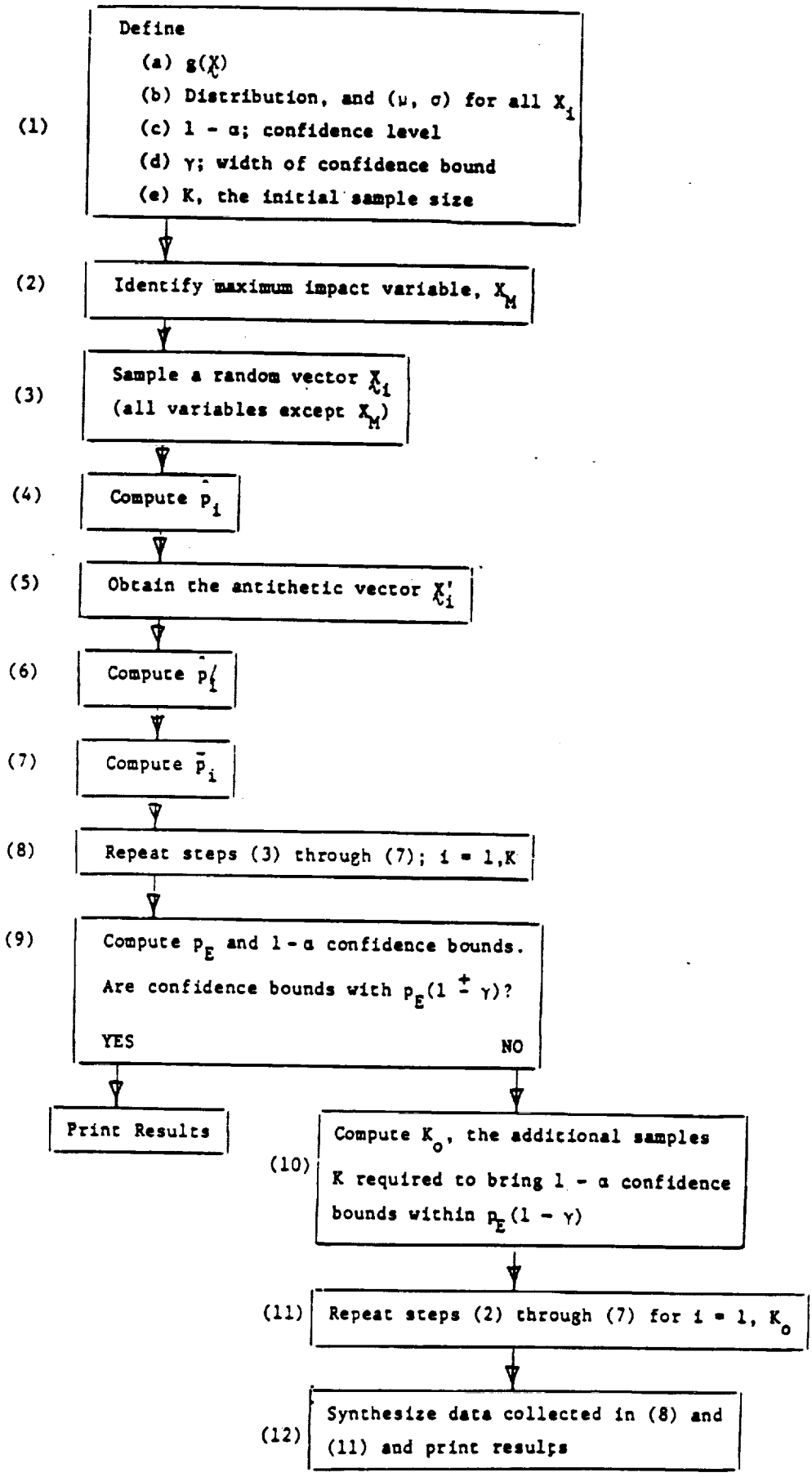


Fig. 3.3 An outline of the variance reduction Monte Carlo program

Fig. 3.4 An example of the output of the variance reduction Monte Carlo Program with commentary

```
                    MONTE CARLO SOLUTION

LIMIT STATE FUNCTION : G=R-DSQRT(300.*P**2+1.92*T**2)

SAMPLE SIZE =    1000
NUMBER OF RANDOM VARIABLES =  3
CONFIDENCE INTERVAL =  95.00 %
GAMMA =      .10

MAX. IMPACT VARIABLE = X( 1)
VARIABLE TYPE IS STRENGTH

                    RANDOM VARIABLES

VARIABLE  DISTRIBUTION        MEAN          STD DEV
    R         WEIBULL      .48000E+02     .30000E+01
    P         LOG          .98700E+00     .16000E+00
    T         EVD          .20000E+02     .20000E+01

ESTIMATE OF P =     .16043E-02
95.00 % CONFIDENCE INTERVALS ARE
PL =     .11725E-02     PU =     .20360E-02

STATISTICS OF P :
MEAN     =    .16043E-02
```

This value is arbitrary; it is the size of the first sample used to estimate the total required sample size, K

Ensures that 95% confidence intervals on p will be within $\pm$ 10% of the estimator, p_E

This is the first estimate of p

Note that 95% confidence intervals exceed $\pm$ 10%. Thus, a larger K is required. (See below)

```
STD DEV =    .69662E-02
MEDIAN  =    .36004E-03
COV     =    .43422E+01

K FOR GAMMA =    .10 IS   7244

ESTIMATE OF P =    .18030E-02
95.00 % CONFIDENCE INTERVALS ARE
PL =    .15509E-02      PU =    .20550E-02

STATISTICS OF P :
MEAN    =    .18348E-02
STD DEV =    .11456E-01
MEDIAN  =    .29017E-03
COV     =    .62436E+01
```

Based on the first sample of K = 10(this is the total K required for th(desired confidence intervals. K is computed from Eq. 3.25 which requir(C_p. This is why the first sample o(1000 is taken.

Note that the confidence intervals do not quit meet the specifications. This is because the original estimate of $C_p = 4.34$ was small relat to the improved estimate of $C_p = 6.24$

DO YOU HAVE ANOTHER DATA SET ?(Y/N)

Note: The size of the sample required K depends upon C_p (Eq. 3.25). In this problem C_p is relatively large implying that a relatively large K is required. This same problem is presented in Table 3.7.

Table 3.1 Example of the Performance of a Variance Reduction Monte Carlo Program; EXAMPLE 1

DEMONSTRATING THE PERFORMANCE OF THE UA VARIANCE REDUCTION MONTE CARLO PROGRAM

EXAMPLE 1

PERFORMANCE FUNCTION: $g = R - S$

Variable	Type	Mean/Median*	Std. Dev./ COV*
R	N.	50.	5.
S	N.	20.	12.

RESULTS:

	Probability of Failure	Total CPU Time(b)	Sample Size, K(c)
Exact(a) Wu/FPI	1.051 E-2		
Monte Carlo Variance Reduction(d)	1.118 E-2	2.04	160
Monte Carlo Conventional (Bernoulli parameter)(e)		11.2	5 E4

Notes:

(a) Exact value using POFAIL if two variables. If more than two, Wu/FPI is used; the exact should be within 5% of this value.

(b) CYBER 175

(c) The number of $\bar{p}_i$ for variance reduction and the number of Z_i for conventional. The values are not directly comparable.

(d) 95% confidence intervals within $\pm$ 10% of p_E

(e) Same confidence interval as variance reduction.

Table 3.2 Example of the Performance of a Variance Reduction Monte Carlo Program; EXAMPLE 2

DEMONSTRATING THE PERFORMANCE OF THE UA VARIANCE REDUCTION MONTE CARLO PROGRAM

EXAMPLE 2

PERFORMANCE FUNCTION: $g = R - S$

Variable	Type	Mean/Median*	Std. Dev./ COV*
R	LN	50. *	0.2 *
S	LN	20. *	0.2 *

RESULTS:

	Probability of Failure	Total CPU-Time(b)	Sample Size, K(c)
Exact(a) Wu/FPI	5.347E-4		
Monte Carlo Variance Reduction(d)	5.072E-4	13.78	11589
Monte Carlo Conventional (Bernoulli parameter)(e)		238.9	1.122 E6

Notes:

(a) Exact value using POFAIL if two variables. If more than two, Wu/FPI is used; the exact should be within 5% of this value.

(b) CYBER 175

(c) The number of $\bar{p}_i$ for variance reduction and the number of Z_i for conventional. The values are not directly comparable.

(d) 95% confidence intervals within $\pm$ 10% of p_E

(e) Same confidence interval as variance reduction.

Table 3.3 Example of the Performance of a Variance Reduction Monte Carlo Program; EXAMPLE 3

DEMONSTRATING THE PERFORMANCE OF THE UA VARIANCE REDUCTION MONTE CARLO PROGRAM

EXAMPLE 3

PERFORMANCE FUNCTION: $g = R - S$

Variable	Type	Mean/Median*	Std. Dev./ COV*
R	WEI.	4.5	0.45
S	FRE.	3.0	0.30

RESULTS:

	Probability of Failure	Total CPU-Time(b)	Sample Size, K(c)
Exact(a) Wu/FPI	1.0933 E-2		
Monte Carlo Variance Reduction(d)	1.0914 E-2	4.066	>535
Monte Carlo Conventional (Bernoulli parameter)(e)		9.634	>5481

Notes:

(a) Exact value using POFAIL if two variables. If more than two, Wu/FPI is used; the exact should be within 5% of this value.

(b) CYBER 175

(c) The number of $\bar{p}_i$ for variance reduction and the number of Z_i for conventional. The values are not directly comparable.

(d) 95% confidence intervals within $\pm$ 10% of p_E

(e) Same confidence interval as variance reduction.

Table 3.4 Example of the Performance of a Variance Reduction Monte Carlo Program; EXAMPLE 4

DEMONSTRATING THE PERFORMANCE OF THE UA VARIANCE REDUCTION MONTE CARLO PROGRAM

EXAMPLE 4

PERFORMANCE FUNCTION: $g = R - S^2$

Variable	Type	Mean/Median*	Std. Dev./ COV*
R	Wei.	20.	4.0
S	Fre.	3.	0.6

RESULTS:

	Probability of Failure	Total CPU-Time(b)	Sample Size, K(c)
Exact(a) Wu/FPI	4.272 E-2		
Monte Carlo Variance Reduction(d)	4.0511 E-2	3.568	1864
Monte Carlo Conventional (Bernoulli parameter)(e)		3.689	9441

Notes:

(a) Exact value using POFAIL if two variables. If more than two, Wu/FPI is used; the exact should be within 5% of this value.

(b) CYBER 175

(c) The number of $\bar{p}_i$ for variance reduction and the number of Z_i for conventional. The values are not directly comparable.

(d) 95% confidence intervals within $\pm$ 10% of p_E

(e) Same confidence interval as variance reduction.

Table 3.5 Example of the Performance of a Variance Reduction Monte Carlo Program; EXAMPLE 5

DEMONSTRATING THE PERFORMANCE OF THE UA VARIANCE REDUCTION MONTE CARLO PROGRAM

EXAMPLE 5

PERFORMANCE FUNCTION: $g = R - S$

Variable	Type	Mean/Median*	Std. Dev./ COV*
R	Nei	20.	2.0
S	EVD	10.	2.0

RESULTS:

	Probability of Failure	Total CPU-Time(b)	Sample Size, K(c)
Exact(a) Wu/FPI	2.85753E-3		
Monte Carlo Variance Reduction(d)	2.6179E-3	10.881	11362
Monte Carlo Conventional (Bernoulli parameter)(e)		36.157	152270

Notes:

(a) Exact value using POFAIL if two variables. If more than two, Wu/FPI is used; the exact should be within 5% of this value.

(b) CYBER 175

(c) The number of $\bar{p}_i$ for variance reduction and the number of Z_i for conventional. The values are not directly comparable.

(d) 95% confidence intervals within $\pm$ 10% of p_E

(e) Same confidence interval as variance reduction.

Table 3.6 Example of the Performance of a Variance Reduction Monte Carlo Program; EXAMPLE 6

DEMONSTRATING THE PERFORMANCE OF THE UA VARIANCE REDUCTION MONTE CARLO PROGRAM

EXAMPLE 6

PERFORMANCE FUNCTION: $g = \frac{\Delta A}{B^m Q} - Ts = \frac{\Delta A}{B^3 \cdot 0.2779} - 6.3E8$

Variable	Type	Mean/Median*	Std. Dev./ COV*
Δ	LN	1.0*	0.3*
A	WEI	4.3E9	0.5*
B	LN	0.9*	0.25*

RESULTS:

	Probability of Failure	Total CPU-Time(b)	Sample Size, K(c)
Exact(a) Wu/FPI	1.901 E-3		
Monte Carlo Variance Reduction(d)	1.7958 E-3	3.643	1437
Monte Carlo Conventional (Bernoulli parameter)(e)		68.3616	199526

Notes:

(a) Exact value using POFAIL if two variables. If more than two, Wu/FPI is used; the exact should be within 5% of this value.

(b) CYBER 175

(c) The number of $\bar{p}_i$ for variance reduction and the number of Z_i for conventional. The values are not directly comparable.

(d) 95% confidence intervals within $\pm$ 10% of p_E

(e) Same confidence interval as variance reduction.

Table 3.7 Example of the Performance of a Variance Reduction Monte Carlo Program; EXAMPLE 7

DEMONSTRATING THE PERFORMANCE OF THE UA VARIANCE REDUCTION MONTE CARLO PROGRAM

EXAMPLE 7

PERFORMANCE FUNCTION: $g = R - \sqrt{300 \cdot P^2 + 1.92 \cdot T^2}$

Variable	Type	Mean/Median*	Std. Dev./ COV*
R	Wei	48.0	3.0
P	LN	0.987*	0.16*
T	EVD	20.0	2.0

RESULTS:

	Probability of Failure	Total CPU-Time(b)	Sample Size, K(c)
Exact(a) Wu/FPI	0.0018		
Monte Carlo Variance Reduction(d)	0.0018208	16.375	12734
Monte Carlo Conventional (Bernoulli parameter)(e)		74.4186	211349

Notes:

(a) Exact value using POFAIL if two variables. If more than two, Wu/FPI is used; the exact should be within 5% of this value.

(b) CYBER 175

(c) The number of $\bar{p}_i$ for variance reduction and the number of Z_i for conventional. The values are not directly comparable.

(d) 95% confidence intervals within $\pm$ 10% of p_E

(e) Same confidence interval as variance reduction.

Table 3.8 Example of the Performance of a Variance Reduction Monte Carlo Program; EXAMPLE 8

DEMONSTRATING THE PERFORMANCE OF THE UA VARIANCE REDUCTION MONTE CARLO PROGRAM

EXAMPLE 8

PERFORMANCE FUNCTION: $g = \Delta - 10000 \cdot \left\{ \frac{f_{pp}}{G(Y \cdot \Delta\epsilon_0)^{-1.71}} + \frac{1 - f_{pp}}{H(Y \cdot \Delta\epsilon_0)^{-1.88}} \right\}$

Variable	Type	Mean/Median*	Std. Dev./ COV*
Δ	LN.	1.0*	0.3*
f_{pp}	N.	0.7	0.07
G	LN.	0.222*	0.4*
Y	LN	1.0*	0.15*
$\Delta\epsilon_0$	EVD.	0.0005	0.00008
H	LN	1.673*	0.4*

RESULTS:

	Probability of Failure	Total CPU-Time(b)	Sample Size, K(c)
Exact(a) Wu/FPI	1.002 E-2		
Monte Carlo Variance Reduction(d)	9.8814E-3	14.822	4401
Monte Carlo Conventional (Bernoulli parameter)(e)		30.7564	≥9810

Notes:

(a) Exact value using POFAIL if two variables. If more than two, Wu/FPI is used; the exact should be within 5% of this value.

(b) CYBER 175

(c) The number of $\bar{p}_i$ for variance reduction and the number of Z_i for conventional. The values are not directly comparable.

(d) 95% confidence intervals within $\pm$ 10% of p_E

(e) Same confidence interval as variance reduction.

Table 3.9 Example of the Performance of a Variance Reduction Monte Carlo Program; EXAMPLE 9a

DEMONSTRATING THE PERFORMANCE OF THE UA VARIANCE REDUCTION MONTE CARLO PROGRAM

EXAMPLE 9a

PERFORMANCE FUNCTION: $g = R - S$

Variable	Type	Mean/Median*	Std. Dev./ COV*
R	LN	20.0*	0.2*
S	LN	10.0*	0.2*

RESULTS:

	Probability of Failure	Total CPU-Time(b)	Sample Size, K(c)
Exact(a) Wu/FPI	6.6642 E-3		
Monte Carlo Variance Reduction(d)	6.4159 E-3	4.75	2831
Monte Carlo Conventional (Bernoulli parameter)(e)		15.7724	59556

Notes:

(a) Exact value using POFAIL if two variables. If more than two, Wu/FPI is used; the exact should be within 5% of this value.

(b) CYBER 175

(c) The number of $\bar{p}_i$ for variance reduction and the number of Z_i for conventional. The values are not directly comparable.

(d) 95% confidence intervals within $\pm$ 10% of p_E

(e) Same confidence interval as variance reduction.

Table 3.10 Example of the Performance of a Variance Reduction Monte Carlo Program; EXAMPLE 9b

DEMONSTRATING THE PERFORMANCE OF THE UA VARIANCE REDUCTION MONTE CARLO PROGRAM

EXAMPLE 9b

PERFORMANCE FUNCTION: $g = R - S$

Variable	Type	Mean/Median*	Std. Dev./ COV*
R	LN	22.5*	0.2*
S	LN	10.0*	0.2*

RESULTS:

	Probability of Failure	Total CPU-Time(b)	Sample Size, K(c)
Exact(a) Wu/FPI	1.89338E-3		
Monte Carlo Variance Reduction(d)	1.7434E-3	8.075	6068
Monte Carlo Conventional (Bernoulli parameter)(e)		51.44	>18776

Notes:

(a) Exact value using POFAIL if two variables. If more than two, Wu/FPI is used; the exact should be within 5% of this value.

(b) CYBER 175

(c) The number of $\bar{p}_i$ for variance reduction and the number of Z_i for conventional. The values are not directly comparable.

(d) 95% confidence intervals within $\pm$ 10% of p_E

(e) Same confidence interval as variance reduction.

Table 3.11 Example of the Performance of a Variance Reduction Monte Carlo Program; EXAMPLE 9c

DEMONSTRATING THE PERFORMANCE OF THE UA VARIANCE REDUCTION MONTE CARLO PROGRAM

EXAMPLE 9c

PERFORMANCE FUNCTION: $g = R - S$

Variable	Type	Mean/Median*	Std. Dev./ COV*
R	LN	>5.0*	0.2*
S	LN	10.0*	0.2*

RESULTS:

	Probability of Failure	Total CPU-Time(b)	Sample Size, K(c)
Exact(a) Wu/FPI	5.347 E-4		
Monte Carlo Variance Reduction(d)	5.072 E-4	13.681	11589
Monte Carlo Conventional (Bernoulli parameter)(e)		164.70	767361

Notes:

(a) Exact value using POFAIL if two variables. If more than two, Wu/FPI is used; the exact should be within 5% of this value.

(b) CYBER 175

(c) The number of $\bar{p}_i$ for variance reduction and the number of Z_i for conventional. The values are not directly comparable.

(d) 95% confidence intervals within $\pm$ 10% of p_E

(e) Same confidence interval as variance reduction.

Table 3.12 Example of the Performance of a Variance Reduction Monte Carlo Program; EXAMPLE 9d

DEMONSTRATING THE PERFORMANCE OF THE UA VARIANCE REDUCTION MONTE CARLO PROGRAM

EXAMPLE 9d.

PERFORMANCE FUNCTION: $g = R - S$

Variable	Type	Mean/Median*	Std. Dev./ COV*
R	LN.	27.0*	0.2*
S	LN.	10.0*	0.2*

RESULTS:

	Probability of Failure	Total CPU-Time(b)	Sample Size, K(c)
Exact(a) Wu/FPI	1.952665E-4		
Monte Carlo Variance Reduction(d)	2.0296E-4	20.27	17977
Monte Carlo Conventional (Bernoulli parameter)(e)		388.93	1840772

Notes:

(a) Exact value using POFAIL if two variables. If more than two, Wu/FPI is used; the exact should be within 5% of this value.

(b) CYBER 175

(c) The number of $\bar{p}_i$ for variance reduction and the number of Z_i for conventional. The values are not directly comparable.

(d) 95% confidence intervals within $\pm$ 10% of p_E

(e) Same confidence interval as variance reduction.

3.8 Comparison of Computer Costs of Variance Reduction and Conventional Monte Carlo

Example 9a, b, c, and d was designed to illustrate how computer costs increase as point probabilities become smaller, providing estimates at the same level of confidence. Figs. 3.5 and 3.6 show the relationship between CYBER 175 CPU execution time and the probability level for the conventional "Bernoulli" and the variance reduction estimates, respectively, for Example 9. Then Fig. 3.7 demonstrates how much more efficient is variance reduction for this problem. It should be noted that Figs. 3.5 through 3.7 relate only to Example 9 and cannot be presented as being characteristic of the relative behavior of the two methods.

3.9 Conclusions on Variance Reduction

Some general conclusions based on experiences exercising VARED are,

1. Variance reduction seems to outperform conventional Monte Carlo consistently. However, in some cases the improvement is dramatic, in some cases it is modest.

2. Related to item 1, it is difficult to predict computer costs. At a given confidence level, CPU time depends strongly upon the form of the performance function, the distribution of the variables, as well as the probability level.

3. To construct a CDF, it is necessary to obtain several point probability estimates, as it is using FPI. Thus, the variance reduction Monte Carlo method is not particularly effective when it is required to construct a distribution function of a response variable.

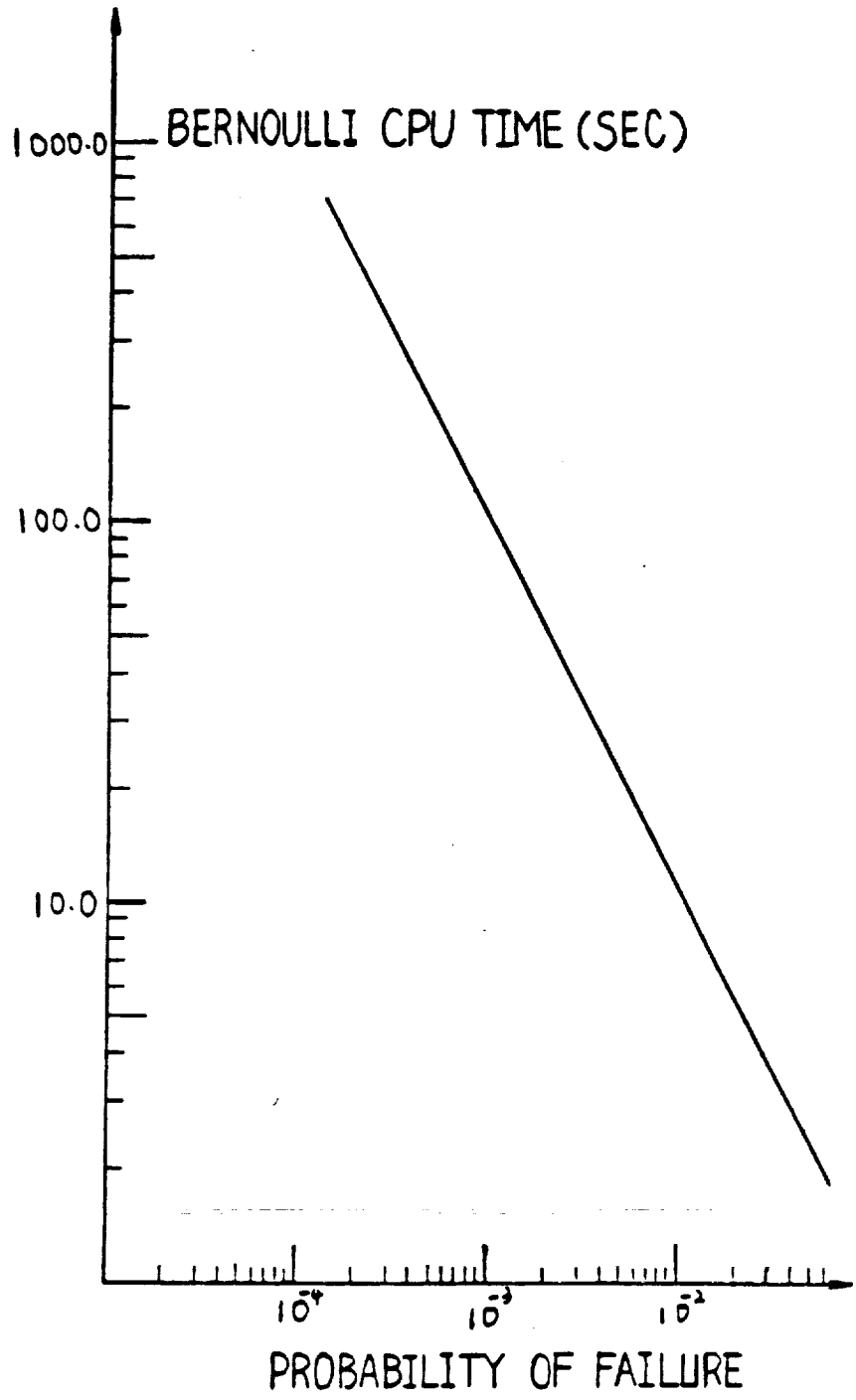


Fig. 3.5 CPU execution time for CYBER 175 for conventional Bernoulli point probability estimate; Example 9; α = 5%, γ = 10%

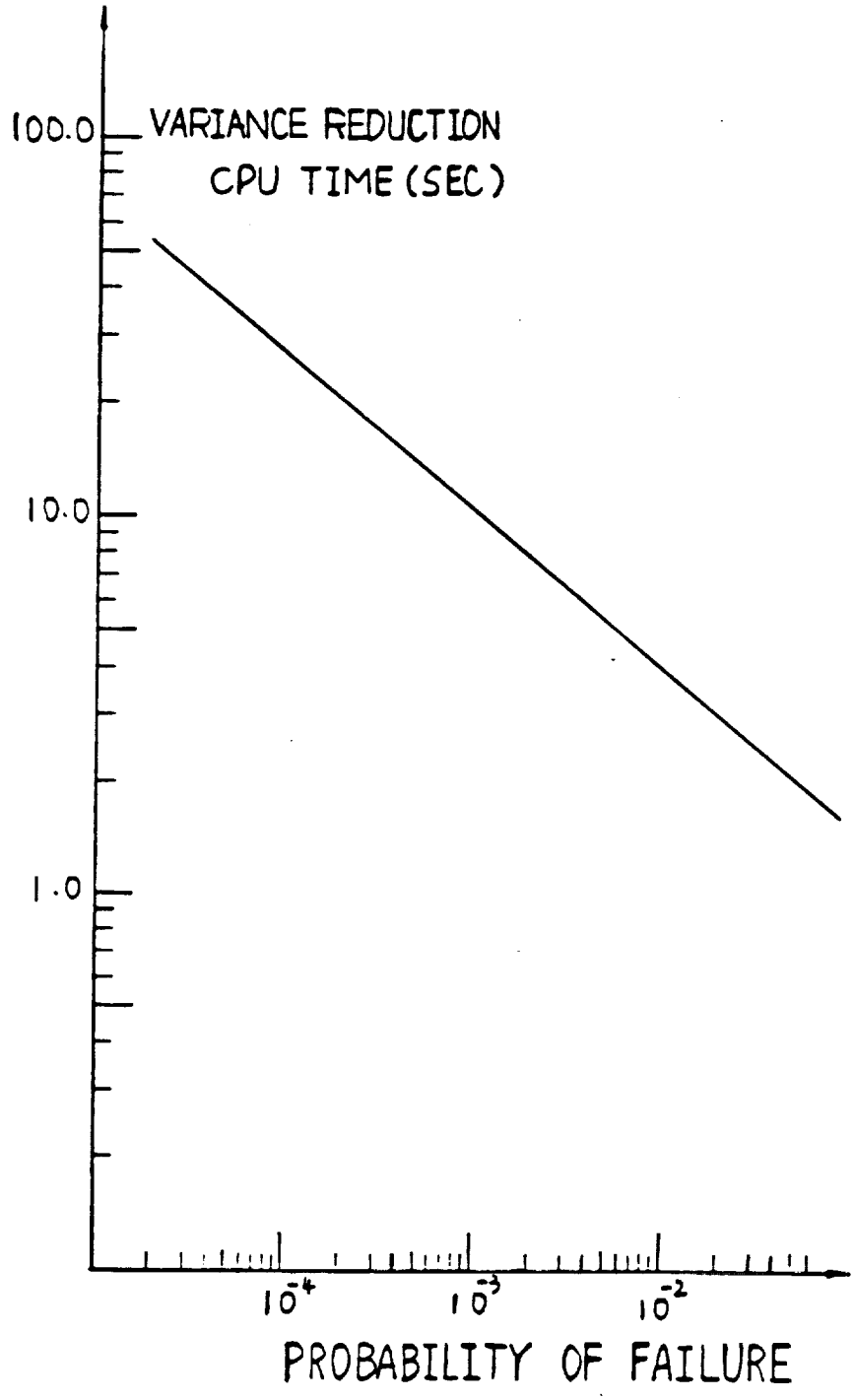


Fig. 3.6 CPU execution time for CYBER 175 for variance reduction point probability estimate; Example 9; α = 5%, γ = 10%

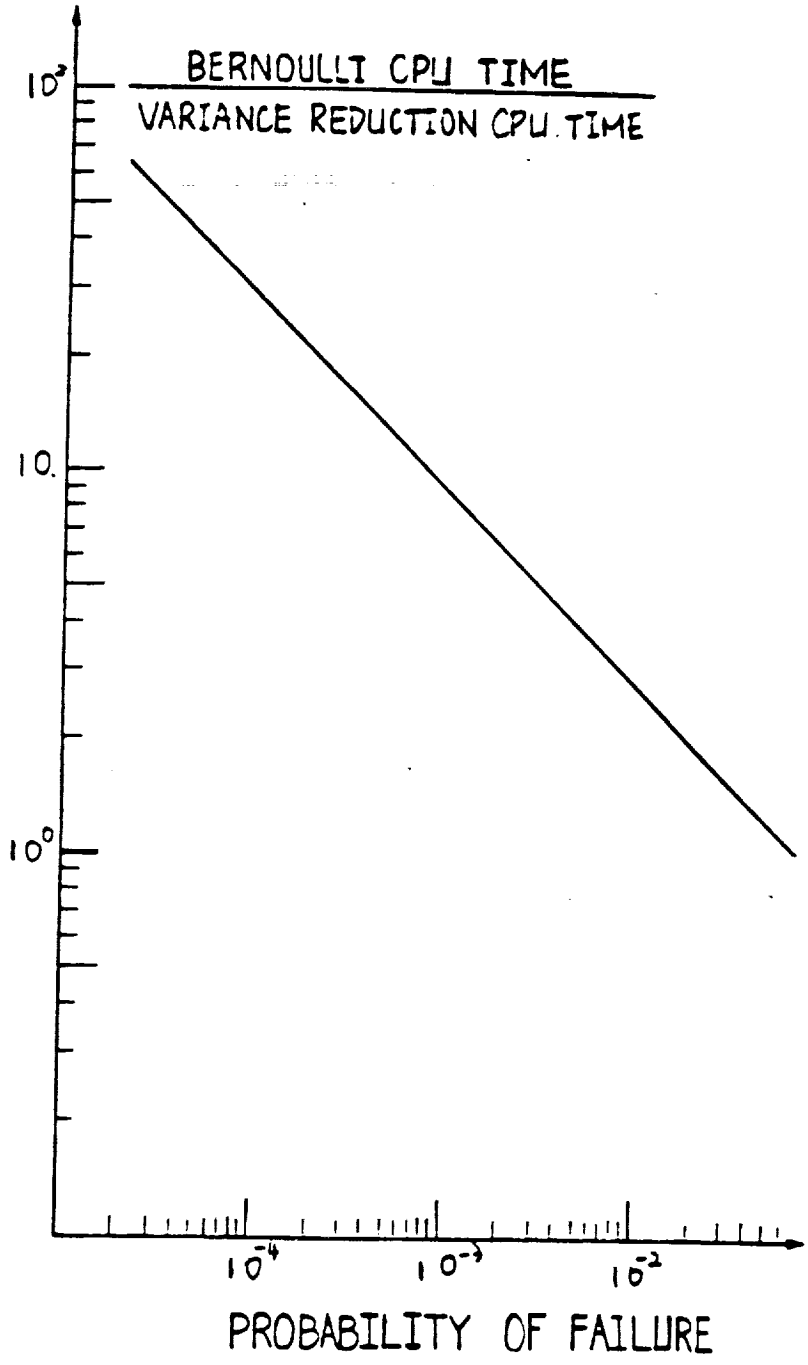


Fig. 3.7 Ratio of Bernoulli to variance reduction CPU execution time for CYBER 175 for point probability estimate; Example 9; $\alpha = 5\%$; $\gamma = 10\%$

APPENDIX A. RANDOM SAMPLES FROM GIVEN DISTRIBUTIONS

Following are the algorithms used to generate random variates from the normal, lognormal, Weibull, extreme value (Type I), and the Frechet distributions. The computer, using a congruential algorithm, samples random numbers u_i from a uniform distribution U(0,1). Forms given below transform uniform variates to variates X_i of other models.

Antithetic variates x_i' (defined as having a negative correlation to x_i) are generated as shown. These antithetic variates are used in the variance reduction method described in Section 3.0.

A. Normal distribution, $N(\mu, \sigma)$; sample two uniform variates, u_i and u_{i+1}. Use the Box-Muller algorithm [1, 2].

$$x_i = \left[\sqrt{-2\ \ell n(u_i)}\ \cos(2\pi\ u_{i+1})\right]\ \sigma + \mu$$

$$x_i' = -x_i + 2\mu$$

B. Lognormal distribution, $LN(\tilde{X}, C_X)$; sample two uniform variates, u_i and u_{i+1}. Use the Box-Muller algorithm [1, 2].

$$\sigma_X = \sqrt{\ell n\ (1 + C_X^2)}$$

$$\mu_X = \ell n\ \tilde{X}$$

$$x_i = \left[\exp(\sqrt{-2\ \ell n(u_i)}\ \cos\ (2\pi\ u_{i+1})\right]\ \sigma_X + \mu_X$$

$$x_i' = \exp(-x_i + 2\ \mu_X)$$

C. Weibull distribution

$$F_X(x) = 1 - \exp\ (\ -\ (\frac{x}{\beta})^{\alpha}\ ) = u \sim U[0,1]$$

$$1 - u = \exp\ (\frac{x}{\beta})^{\alpha}\ ) \sim\ U[0,1]$$

$$-\ \ell n\ (1 - u) = (\frac{x}{\beta})^{\alpha}$$

Thus,

$$x_i = \beta(-\ \ell n\ (1 - u_i)^{1/\alpha}$$

$$x_i' = \beta(-\ \ell n\ (u_i))^{1/\alpha}$$

D. EVD distribution

$$F_X(x) = \exp\left(-\exp(-\alpha(X - \beta)\right) = u \sim U[0,1]$$

$$\exp\left(-\alpha(x - \beta)\right) = -\ln u$$

$$-\alpha(x - \beta) = \ln\left(-\ln u\right)$$

Thus,

$$x_i = \beta - \frac{1}{\alpha}\ln(-\ln(u_i))$$

$$x_i' = \beta - \frac{1}{\alpha}\ln(-\ln(1 - u_i))$$

E. Frechet distribution

$$F_X(x) = \exp -\left(\frac{v}{x}\right)^k = u \sim U[0,1]$$

$$\left(\frac{v}{x}\right)^k = -\ln(u)$$

Thus,

$$x_i = v\ (-\ln(u_i))^{-1/k}$$

$$x_i' = v\ (-\ln(1 - u_i))^{-1/k}$$

APPENDIX B. LISTING OF CONVENTIONAL MONTE CARLO PROGRAM (COMOC)

This version runs on the VAX and the CYBER 175. It is not interactive.

The performance function is introduced in subroutine LSFMC as XA.

See listing.

Card 1 Limit state function (not used in program; only printed on output)

Card 2 Number of trials; number of variables (free format)

Card 3 PLOT and ISTD type

PLOT: Y_i's are sorted to construct empirical CDF

0 = no sort

1 = Y_i's are sorted

ISTD; option to enter standard deviations or coefficients of deviations or coefficients of variation of each variable (if lognormal, always use COV).

0 = COV

1 = Std. dev.

Now enter each variable, its distribution type, and its moments.

Card 4 Variable name.

Card 5 Distribution, mean, and standard deviation

1 = WEI (Weibull)

2 = NORM (Normal

3 = EVD (Extreme value distribution)

4 = LN (Lognormal; always use median and COV)

5 = FRE (Frechet)

Then repeat 4 and 5 for all of the other variables.

```
      PROGRAM GMC
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION INAME(40),XMEAN(40),XSTD(40),DIST(40),DTRANS(40),X
      DIMENSION Y(30000),F(5),AL(40),BE(40)
        COMMON /TWO/ PI,PI2
        CHARACTER*80 GRS,FIN,FOUT,DTRANS*7,AA*7,BB*6,CC*3,DD*3,EE*
        DATA AA/'WEIBULL'/
        DATA BB/'NORMAL'/
        DATA CC/'EVD'/
        DATA DD/'LOG'/
        DATA EE/'FRECHET'/
651     FORMAT(A10)
8004    CONTINUE
        READ(5,'(A)',END=8888) GRS
        READ(5,*) K,N
        READ(5,*) PLOT1,PLOT2,ISTD
        READ(5,*) ISEED
        DO 7901 I=1,N
        READ(5,651) INAME(I)
        READ(5,*) DIST(I),XMEAN(I),XSTD(I)
 7901   CONTINUE
        IF(ISTD.EQ.0) THEN
        DO 913 I=1,N
        IF(DIST(I).EQ.4.) GO TO 913
        XSTD(I)=XMEAN(I)*XSTD(I)
913     CONTINUE
        END IF
C
C
        DO 1234 I=1,N
        AL(I)=0.D0
        BE(I)=0.D0
1234    CONTINUE
        PI=4.D0*DATAN(1.D0)
        PI2=PI+PI
      DO 1 I=1,N
      IF(DIST(I).EQ.1.) DTRANS(I)=AA
      IF(DIST(I).EQ.2.) DTRANS(I)=BB
      IF(DIST(I).EQ.3.) DTRANS(I)=CC
      IF(DIST(I).EQ.4.) DTRANS(I)=DD
      IF(DIST(I).EQ.5.) DTRANS(I)=EE
        IF(DIST(I).EQ.1.) CALL WEI(XMEAN(I),XSTD(I),AL(I),BE(I))
        IF(DIST(I).EQ.3.) CALL EVD(XMEAN(I),XSTD(I),AL(I),BE(I),PI)
        IF(DIST(I).EQ.5.) CALL FRE(XMEAN(I),XSTD(I),AL(I),BE(I))
    1 CONTINUE
C
C*  THE DATA IS PRINTED OUT.
C
      WRITE(6,11) GRS,K,N
      WRITE(6,12)
      WRITE(6,13)(INAME(I),DTRANS(I),XMEAN(I),XSTD(I),I=1,N)
C       GENERATE RANDOM # AND CORRESPONDING RANDOM VARIABLE
        NUM=0
      DO 4 I=1,K
      DO 3 J=1,N
      CALL GENX(DIST(J),AL(J),BE(J),X(J),XMEAN(J),XSTD(J),ISEED)
    3   CONTINUE
        CALL LSFMC(Y(I),N,X)
        IF(Y(I).LE.0.D0) NUM=NUM+1
```

CONVENTIONAL MONTE CARLO PROGRAM (COMOC): Runs on the VAX or CYBER 175

```
      4 CONTINUE
    123 CALL STAT(Y,K,YMEAN,YSTD,YMED,YCOV)
    124 WRITE(6,15) YMEAN,YSTD,YMED,YCOV
C
C*   ROUTINE TO ACCUMULATE NUMBER OF TRIALS WITH NEGATIVE Y(I)
C*   VALUES AND PRINT OUT RESULTS
C
       RATIO = DBLE(NUM)/DBLE(K)
       WRITE(6,9) NUM,RATIO
     9 FORMAT(/,10X,'NUMBER OF NEG Y VALUES =',I5,'.',4X,
      +'PERCENT OF TRIALS=',F9.6)
C
C*   THE SORTED VALUE OF Y AND THE EMPIRICAL CDF ARE PRINTED.
          IF(PLOT1.EQ.0.) GO TO 92
          CALL QSORT(Y,K)
92        IF(PLOT2.EQ.0.) GO TO 3456
          WRITE(6,1017)
1017      FORMAT(////,14X,'SORTED VALUES OF Y AND THE EMPIRICAL CDF',
          J1=1
          J2=5
3030      WRITE(6,1003) J1,(Y(I),I=J1,J2)
1003      FORMAT(1X,'I = ',I5,5E13.5)
          J1=J1+5
          J2=J2+5
          IF(J1.GT.K) GO TO 3031
          IF(J2.GT.K) THEN
          J2=K
          GO TO 3030
          END IF
          GO TO 3030
3031      CONTINUE
          WRITE(6,67)
67        FORMAT(/)
          J=0
          J1=1
          DO 1009 I=1,K
          J=J+1
          F(J)=(DBLE(I)-.5)/DBLE(K)
          IF(J.EQ.5.OR.I.EQ.K) THEN
          WRITE(6,1003) J1,(F(L),L=1,J)
          J=0
          J1=J1+5
          END IF
1009      CONTINUE
3456      CONTINUE

   11 FORMAT(5(/),30X,'MONTE CARLO SOLUTION',5(/),10X,
     +'LIMIT STATE FUNCTION : ',A,5(/),10X,
     +'SAMPLE SIZE, K=',I7//10X,'NUMBER OF RANDOM VARIABLES, N=',I3//
   12 FORMAT(26X,'RANDOM VARIABLES',//10X,'VARIABLE',2X,
     +'DISTRIBUTION',8X,'MEAN',12X,'STD DEV')
   13 FORMAT(/11X,A7,5X,A7,5X,E12.5,5X,E12.5)
   15 FORMAT(/////10X,'STATISTICS OF Y :'//10X,'MEAN    =',E13.5//10X
     +'STD DEV =',E13.5//10X,'MEDIAN  =',E13.5//10X,'COV     =',
     +E13.5,4(/))
   17 FORMAT(1H1,2(/),14X,'SORTED VALUES OF Y AND THE EMPIRICAL CDF')
   19 FORMAT((5E13.5))
         GO TO 8004
8888     CONTINUE
  125 STOP
```

```
      END
      SUBROUTINE STAT(U,M,XM,STD,XMED,COV)
C
C*  THIS SUBROUTINE CALCULATES THE STATISTICS (MEAN,STD DEV,MEDIAN
C*  OF Y FUNCTION.
C
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION U(M)
      XK=M
      XM=0.
      DO 63 I=1,M
      XM=XM+U(I)
   63 CONTINUE
      XM=XM/XK
      STD=0.
      DO 64 I=1,M
      STD=STD+(U(I)-XM)**2
   64 CONTINUE
      STD=STD/(XK-1.D0)
      STD=DSQRT(STD)
      COV=STD/XM
      XMED=XM/DSQRT(1.D0+COV**2)
      RETURN
      END
        SUBROUTINE GENX(DIST,ALPHA,BETA,X,XMEAN,XSTD,ISEED)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        COMMON /TWO/ PI,PI2
C
        IDIST=INT(DIST+.1)
        AA=RAN(ISEED)
        GO TO (1,2,3,4,5), IDIST
 1      X=BETA*(-DLOG(AA))**(1.D0/ALPHA)
        RETURN
 2      BB=RAN(ISEED)
        E=DSQRT(-2.D0*DLOG(AA))
        X=E*DCOS(PI2*BB)*XSTD+XMEAN
        RETURN
 3      X=BETA-DLOG(-DLOG(AA))/ALPHA
        RETURN
 4      BB=RAN(ISEED)
        SDX=DSQRT(DLOG(1.D0+XSTD**2))
        UX=DLOG(XMEAN)
        E=DSQRT(-2.D0*DLOG(AA))
        X=DEXP(E*DCOS(PI2*BB)*SDX+UX)
        RETURN
 5      X=BETA*(-DLOG(AA))**(-1.D0/ALPHA)
        RETURN
        END
        SUBROUTINE BISECT(COV,ISIGN,ALPHA)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C       ISIGN = 1; WEIBULL DIST.
C             = 2; FRECHET DIST.
        F(X,COV)=-(1.D0+COV**2)*GAMMA(X)**2+GAMMA(2.*X)
        IF(ISIGN.EQ.1) X1=COV**(1.08)
        IF(ISIGN.EQ.2) X1=COV**(.677)/2.33
        IF(ISIGN.EQ.2.AND.X1.GT..49D0) X1=.48999999
 7      IF(ISIGN.EQ.1) F1=F(X1,COV)
        IF(ISIGN.EQ.2) F1=F(-X1,COV)
        IF(DABS(F1).LE.1.D-10) GO TO 1
        X2=X1+.01D0
```

GENX obtains random samples from distributions

RAN is library uniform random number generator for CYBER 175

```
      IF(ISIGN.EQ.1) F2=F(X2,COV)
      IF(ISIGN.EQ.2) F2=F(-X2,COV)
      F12=F1*F2
      IF(F12.LT.0.) GO TO 20
      IF(DABS(F1).GT.DABS(F2)) X1=X2
      IF(DABS(F1).LT.DABS(F2)) X1=X1-.01D0
      GO TO 7
20    CONTINUE
2     X3=(X1+X2)*.5D0
      IF(ISIGN.EQ.1) F13=F(X1,COV)*F(X3,COV)
      IF(ISIGN.EQ.2) F13=F(-X1,COV)*F(-X3,COV)
      IF(F13.LT.0.) X2=X3
      IF(F13.GT.0.) X1=X3
      DX=DABS(X1-X2)
      IF(DX.GE.1.D-9) GO TO 2
1     ALPHA=1.D0/X1
      RETURN
      END
      SUBROUTINE WEI(XMEAN,XDEV,ALPHA,BETA)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COV=XDEV/XMEAN
      CALL BISECT(COV,1,ALPHA)
      AL1=1.D0/ALPHA
      BETA=XMEAN/GAMMA(AL1)
      RETURN
      END
      SUBROUTINE FRE(XMEAN,XDEV,ALPHA,BETA)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COV=XDEV/XMEAN
      CALL BISECT(COV,2,ALPHA)
      AL1=1.D0/ALPHA
      BETA=XMEAN/GAMMA(-AL1)
      RETURN
      END
      SUBROUTINE EVD(XMEAN,STD,ALPHA,BETA,PI)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      ALPHA=PI/(STD*DSQRT(6.D0))
      BETA=XMEAN-.577215664901533/ALPHA
      RETURN
      END
      DOUBLE PRECISION FUNCTION GAMMA(Y1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /TWO/ PI,PI2
      X=Y1+1.D+0
      Z=X
      IF(X.GE.6.0D+0)GO TO 456
      N=INT(X)
      Z=(6.0D+0)-N+X
456   Y=1.D+0/Z**2
      ALG=(Z-.5D+0)*DLOG(Z)+.5D+0*DLOG(PI2)-
     $ Z-(1.D+0/(12.D+0*Z))*(((Y/0.14D+3-1.D+0/0.105D+3)*Y+
     $ 1.D+0/.3D+2)*Y-1.D+0)
      IF(X.GE.6.D+0)GO TO 457
      ITE=6-N
      DO 3 J=1,ITE
      A=X+J-1.D+0
      ALG=ALG-DLOG(A)
3     CONTINUE
457   GAMMA=DEXP(ALG)
      RETURN
```

BISECT used to compute Weibull and Frechet shape parameter (exponent)

Computes Weibull parameters

Computes Frechet parameters

Computes EVD parameters

The gamma function

```
240:            END
241:            SUBROUTINE QSORT(A,N)
242:            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
243:            DIMENSION A(N),KSL(24),KSR(24)
244:            KS=1
245:            KSL(1)=1
246:            KSR(1)=N
247:  10        CONTINUE
248:            L=KSL(KS)
249:            KR=KSR(KS)
250:            KS=KS-1
251:  20        CONTINUE
252:            I=L
253:            J=KR
254:            LR=(L+KR)/2
255:            X=A(LR)
256:  30        CONTINUE
257:            IF(A(I).LT.X) THEN
258:            I=I+1
259:            GO TO 30
260:            END IF
261:  40        CONTINUE
262:            IF(X.LT.A(J)) THEN
263:            J=J-1
264:            GO TO 40
265:            END IF
266:            IF(I.LE.J) THEN
267:            W=A(I)
268:            A(I)=A(J)
269:            A(J)=W
270:            I=I+1
271:            J=J-1
272:            END IF
273:            IF(I.LE.J) GO TO 30
274:            IF(I.LT.KR) THEN
275:            KS=KS+1
276:            KSL(KS)=I
277:            KSR(KS)=KR
278:            END IF
279:            KR=J
280:            IF(L.LT.KR) GO TO 20
281:            IF(KS.NE.0) GO TO 10
282:            RETURN
283:            END
284:            SUBROUTINE LSFMC(XA,N,X)
285:            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
286:            DIMENSION X(N)
287:            XA=X(1)-X(2)
288:            RETURN
289:            END
*
```

This is the sort routine, QUICKSORT

This is where the limit state is introduced

APPENDIX C. THE SORT ROUTINE: "QUICKSORT"

QUICKSORT is described in detail in the book by Wirth [7], who describes its performance as "spectacular," and claims that it is the best sorting method on arrays known so far. The method is based on exchanges and the inventor C.A.R.Hoare recognized that sorting becomes most efficient when exchanges are made over large distances.

The table below shows execution times (in milliseconds) consumed by several proposed sorting methods as executed by the PASCAL system on a CDC 6400 computer. The three columns contain times used to sort the already ordered array, a random permutation, and the inversely ordered array. The left figure in each column is for 256 items, and right one for 512 items.

In summary, the computational effort needed for QUICKSORT is of the order of n log n.

	Ordered		Random		Inversely Ordered	
Straight insertion	12	23	366	1444	704	2836
Binary insertion	56	125	373	1327	662	2490
Straight selection	489	1907	509	1956	695	2675
Bubblesort	540	2165	1026	4054	1492	5931
Bubblesort with flag	5	8	1104	4270	1645	6542
Shakersort	5	9	961	3642	1619	6520
Shellsort	58	116	127	349	157	492
Heapsort	116	253	110	241	104	226
Quicksort	31	69	60	146	37	79
Mergesort	99	234	102	242	99	232

Execution Times of Sort Programs.

APPENDIX D. LISTING OF THE VARIANCE REDUCTION MONTE CARLO PROGRAM (VARED)

This version runs on the VAX and the CYBER 175. It is not interactive.

The performance function is introduced in subroutine LSFMC, then compiled and linked to the rest of the program.

Data Input for the VAX Version Variance Reduction Program

Card 1 Limit State Function (not used for calculations in the program)

Ex: g = R - S or R = S, etc.

Card 2 Number of Trials (the preliminary value of K); Number of Variables; Maximum Error in Secant Method for Solution of Maximum Impact Variable (a small number)

Ex: 1000, 3, 1.D-6

or 10000,5,1.D-7

Card 3 Confidence Interval; Gamma; ISTD;

a. C.I. = 0 to 100 in percent: Ex: 90; implies 90% C.I.

b. Gamma $0 \leq \gamma \leq 1$, but typically choose γ from 0.05 to 0.20. See Eq. 3.21 ff.

c. ISTD = OPTION to enter standard derivations and coefficients of variation of each variable (for LN Dist, always use COV)

0 = COV

1 = Std. dev.

Card 4 Enter ISEED

Any integer number between 0 and 262,139 to start the random sampling.

Ex: 23, 579, etc.

Card 5 Enter variable name. (Free format)

Card 6 Enter corresponding distribution, mean, and standard deviation (if LN always input median and COV); Ex: 1, 20, 2

a. dist. = 1 = Weibull

2 = Normal

3 = EVD

4 = Lognormal (LN)

5 = Frechet

Then repeat 5 and 6 for all of the other variables.

```
1:*       PROGRAM GMC
2:          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
3:        DIMENSION INAME(20),XMEAN(20),XSTD(20),DIST(20),DTRANS(20),X(
4:        DIMENSION Y(10000),F(5),AL(20),BE(20),XA(20),TX(20),TS(20)
5:          COMMON /TWO/PI,SPI2,PI2
6:          CHARACTER*70 GRS,FIN,FOUT,AA*7,BB*6,CC*3,DD*3,EE*7
7:          CHARACTER*7 INAME,DTRANS
8:          DATA AA/'WEIBULL'/
9:          DATA BB/'NORMAL'/
10:         DATA CC/'EVD'/
11:         DATA DD/'LOG'/
12:         DATA EE/'FRECHET'/
13: C
14:         READ(5,'(A)',END=8888) GRS
15:         READ(5,*) K,N,EPS
16:         READ(5,*) ZAL,GAM,ISTD,PLOT
17:         READ(5,*) ISEED
18:         DO 7901 I=1,N
19:         READ(5,'(A)') INAME(I)
20:         READ(5,*) DIST(I),XMEAN(I),XSTD(I)
21: 7901    CONTINUE
22: 8004    CONTINUE
23:         IF(ISTD.EQ.0) THEN
24:         DO 913 I=1,N
25:         IF(DIST(I).EQ.4.) GO TO 913
26:         XSTD(I)=XMEAN(I)*XSTD(I)
27: 913     CONTINUE
28:         END IF
29:       IF(K.GT.10000) K=10000
30: C
31: C
32:         DO 1234 I=1,N
33:         AL(I)=0.D0
34:         BE(I)=0.D0
35:         IF(DIST(I).EQ.4.) THEN
36:         TX(I)=XMEAN(I)*DSQRT(1.D0+XSTD(I)**2)
37:         TS(I)=TX(I)*XSTD(I)
38:         ELSE
39:         TX(I)=XMEAN(I)
40:         TS(I)=XSTD(I)
41:         END IF
42: 1234    CONTINUE
43:         PI=4.D0*DATAN(1.D0)
44:         PI2=PI+PI
45:         SPI2=1.D0/DSQRT(PI2)
46:       DO 1 I=1,N
47:       IF(DIST(I).EQ.1.) DTRANS(I)=AA
48:       IF(DIST(I).EQ.2.) DTRANS(I)=BB
49:       IF(DIST(I).EQ.3.) DTRANS(I)=CC
50:       IF(DIST(I).EQ.4.) DTRANS(I)=DD
51:       IF(DIST(I).EQ.5.) DTRANS(I)=EE
52:         IF(DIST(I).EQ.1.) CALL WEI(XMEAN(I),XSTD(I),AL(I),BE(I))
53:         IF(DIST(I).EQ.3.) CALL EVD(XMEAN(I),XSTD(I),AL(I),BE(I),PI)
54:         IF(DIST(I).EQ.5.) CALL FRE(XMEAN(I),XSTD(I),AL(I),BE(I))
55:     1 CONTINUE
56: C
57: C*  THE DATA IS PRINTED OUT.
58: C
59: C       MAIN LOOP USING ANTITHETIC VARIANCE REDUCTION METHOD
```

Program VARED. Monte Carlo using variance reduction method; runs on the VAX or CYBER 175

```
      C        FIND MAX. IMPACT VARIABLE
               DG=0.D0
               CALL LSFMC(G,N,TX)
               DO 700 I=1,N
               TX(I)=TX(I)+TS(I)
               CALL LSFMC(DGB,N,TX)
               DGA=DGB-G
               IF(DABS(DGA).LE.DABS(DG)) GO TO 701
               IV=I
               DG=DGA
      701      TX(I)=TX(I)-TS(I)
      700      CONTINUE
      C
             WRITE(6,11) GRS,K,N
               WRITE(6,96) ZAL,GAM
      96       FORMAT(10X,'CONFIDENCE INTERVAL = ',F6.2,' %',//,
     $       10X,'GAMMA = ',F6.2,///)
               WRITE(6,559) IV
      559      FORMAT(10X,'MAX. IMPACT VARIABLE = X(',I2,')',/)
               IF(DG.LE.0.D0) WRITE(6,561)
      561      FORMAT(10X,'VARIABLE TYPE IS STRESS',///)
               IF(DG.GT.0.D0) WRITE(6,563)
      563      FORMAT(10X,'VARIABLE TYPE IS STRENGTH',///)
             WRITE(6,12)
             WRITE(6,13)(INAME(I),DTRANS(I),XMEAN(I),XSTD(I),I=1,N)
      C        CALCULATE PROB. OF FAILURE
               K1=1
               K2=K
               ICO=1
      98       CONTINUE
               DO 702 I=K1,K2
               DO 703 J=1,N
               IF(J.EQ.IV) GO TO 703
               CALL GENX(DIST(J),AL(J),BE(J),X(J),XA(J),XMEAN(J),XSTD(J),ISE
      703      CONTINUE
               IF(DG.GT.0.D0) A=TX(IV)-3.D0*TS(IV)
               IF(DG.LE.0.D0) A=TX(IV)+2.D0*TS(IV)
               B=A+TS(IV)
               CALL SECA(EPS,A,B,IV,N,X)
               CALL CDFPDF(DIST(IV),AL(IV),BE(IV),X(IV),XMEAN(IV),XSTD(IV),
     $       1,CDF1,PDF)
               IF(DG.LE.0.D0) CDF1=1.D0-CDF1
               IF(DG.GT.0.D0) A=TX(IV)-3.D0*TS(IV)
               IF(DG.LE.0.D0) A=TX(IV)+2.D0*TS(IV)
               B=A+TS(IV)
               CALL SECA(EPS,A,B,IV,N,XA)
               CALL CDFPDF(DIST(IV),AL(IV),BE(IV),XA(IV),XMEAN(IV),XSTD(IV),
     $       1,CDF2,PDF)
               IF(DG.LE.0.D0) CDF2=1.D0-CDF2
               Y(I)=(CDF1+CDF2)*.5D0
      702      CONTINUE
      C
         123 CALL STAT(Y,K1,K2,YMEAN,YSTD,YMED,YCOV)
               IF(ICO.EQ.1) THEN
               YM=YMEAN
               YS=YSTD
               YME=YMED
               YC=YCOV
               YM1=YM
               ELSE
```

```
      YM=(K*YM1+(K2-K)*YMEAN)/K2
      YS1=YS**2*(K-1)+K*YM1**2+YSTD**2*(K2-K-1)+(K2-K)*YMEAN**2
      YS2=YS1-K2*YM**2
      YS=DSQRT(YS2/(K2-1))
      YC=YS/YM
      YME=YM/DSQRT(1.D0+YC**2)
      END IF
      ZAL1=.005D0*(100.D0+ZAL)
      ZAX=XINV(ZAL1)
      ZX=ZAX*YC/DSQRT(DBLE(K2))
      PL=YM*(1.D0-ZX)
      PU=YM*(1.D0+ZX)
      WRITE(6,176) YM,ZAL,PL,PU
176   FORMAT(///,10X,'ESTIMATE OF P = ',E13.5,//,
     $ 10X,F5.2,' % CONFIDENCE INTERVALS ARE',//,
     $ 10X,'PL = ',E13.5,5X,'PU = ',E13.5,///)
C
      WRITE(6,15) YMEAN,YSTD,YMED,YCOV
     IF(PLOT.EQ.0.) GO TO 3456
     CALL QSORT(Y,K2)
C*   THE SORTED VALUE OF Y AND THE EMPIRICAL CDF ARE PRINTED.
C
      WRITE(6,1017)
1017  FORMAT(1H1,//,14X,'SORTED VALUES OF Y AND THE EMPIRICAL CDF',
      J1=1
      J2=5
3030  WRITE(6,1003) J1,(Y(I),I=J1,J2)
1003  FORMAT(1X,'I = ',I5,5E13.5)
      J1=J1+5
      J2=J2+5
      IF(J1.GT.K2) GO TO 3031
      IF(J2.GT.K2) THEN
      J2=K2
      GO TO 3030
      END IF
      GO TO 3030
3031  CONTINUE
      WRITE(6,67)
67    FORMAT(/)
      J=0
      J1=1
      DO 1009 I=1,K2
      J=J+1
      F(J)=(DBLE(I)-.5)/DBLE(K2)
      IF(J.EQ.5.OR.I.EQ.K2) THEN
      WRITE(6,1003) J1,(F(L),L=1,J)
      J=0
      J1=J1+5
      END IF
1009  CONTINUE
3456  CONTINUE
      K1=K+1
      K2=(YC*ZAX/GAM)**2+1
      IF(ICO.EQ.1) WRITE(6,99) GAM,K2
99    FORMAT(//,10X,'K FOR GAMMA = ',F6.2,' IS ',I6)
      ICO=ICO+1
      IF(ICO.EQ.2.AND.K2.GT.K) GO TO 98
   11 FORMAT(1H1,5(/),30X,'MONTE CARLO SOLUTION',5(/),10X,
     +'LIMIT STATE FUNCTION : ',A,5(/),10X,
     +'SAMPLE SIZE =',I7//10X,'NUMBER OF RANDOM VARIABLES =',I3//)
```

```
180:     12 FORMAT(26X,'RANDOM VARIABLES',//10X,'VARIABLE',2X,
181:       +'DISTRIBUTION',8X,'MEAN',12X,'STD DEV')
182:     13 FORMAT(/11X,A7,5X,A7,5X,E12.5,5X,E12.5)
183:     15 FORMAT(/////10X,'STATISTICS OF P :'//10X,'MEAN     =',E13.5//10X,
184:       +'STD DEV =',E13.5//10X,'MEDIAN  =',E13.5,//10X,'COV      =',
185:       +E13.5,////)
186:           IF(ANS1.EQ.'F'.OR.ANS1.EQ.'f') GO TO 8300
187:           WRITE(6,8301)
188: 8301      FORMAT(' DO YOU HAVE ANOTHER DATA SET ?(Y/N) ',$)
189:           READ(5,8001) ANS3
190:           IF(ANS3.EQ.'Y'.OR.ANS3.EQ.'y') GO TO 8304
191: 8888      CONTINUE
192:    125 STOP
193:        END
194:           SUBROUTINE QSORT(A,N)
195:           IMPLICIT DOUBLE PRECISION (A-H,O-Z)
196:           DIMENSION A(N),KSL(240),KSR(240)
197:           KS=1
198:           KSL(1)=1
199:           KSR(1)=N
200:  10       CONTINUE
201:           L=KSL(KS)
202:           KR=KSR(KS)
203:           KS=KS-1
204:  20       CONTINUE
205:           I=L
206:           J=KR
207:           LR=(L+KR)/2
208:           X=A(LR)
209:  30       CONTINUE
210:           IF(A(I).LT.X) THEN
211:           I=I+1
212:           GO TO 30
213:           END IF
214:  40       CONTINUE
215:           IF(X.LT.A(J)) THEN
216:           J=J-1
217:           GO TO 40
218:           END IF
219:           IF(I.LE.J) THEN
220:           W=A(I)
221:           A(I)=A(J)
222:           A(J)=W
223:           I=I+1
224:           J=J-1
225:           END IF
226:           IF(I.LE.J) GO TO 30
227:           IF(I.LT.KR) THEN
228:           KS=KS+1
229:           KSL(KS)=I
230:           KSR(KS)=KR
231:           END IF
232:           KR=J
233:           IF(L.LT.KR) GO TO 20
234:           IF(KS.NE.0) GO TO 10
235:           RETURN
236:           END
237:           SUBROUTINE SECA(EPS,A,B,IV,N,X)
238:           IMPLICIT DOUBLE PRECISION (A-H,O-Z)
239:           DIMENSION X(N)
```

Sort routine used originally for program development. Not needed for operational version.

```
240:            X(IV)=A
241:            CALL LSFMC(U,N,X)
242:            X(IV)=B
243:            CALL LSFMC(V,N,X)
244:  1         CONTINUE
245:            IF(DABS(X(IV)-A).GE.EPS) THEN
246:            X(IV)=B-V*(B-A)/(V-U)
247:            A=B
248:            B=X(IV)
249:            U=V
250:            CALL LSFMC(V,N,X)
251:            GO TO 1
252:            END IF
253:            RETURN
254:            END
255:            SUBROUTINE CDFPDF(DIST,ALPHA,BETA,X,XMEAN,XDEV,ICDF,CDF,PDF)
256:            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
257:            COMMON /TWO/PI,SPI2,PI2
258:            IDIST=INT(DIST+.1)
259:          GO TO (1,2,3,4,5),IDIST
260:        1 RB=X/BETA
261:          EW=RB**ALPHA
262:          IF(EW.GT.200.) EW=200.
263:          EXPWEI=DEXP(-EW)
264:          CDF=1.D0-EXPWEI
265:          IF(ICDF.EQ.1) GO TO 10
266:          PDF=(ALPHA/BETA)*(EW/RB)*EXPWEI
267:          GO TO 10
268:        2 Z=(X-XMEAN)/XDEV
269:          CDF=CDFNOR(Z)
270:          IF(ICDF.EQ.1) GO TO 10
271:          PDF=SPI2*DEXP(-Z**2*.5D0)/XDEV
272:          GO TO 10
273:        3 EE=ALPHA*(X-BETA)
274:          IF(EE.GT.200.) EE=200.
275:          YY=DEXP(-EE)
276:          IF(YY.GT.200.) YY=200.
277:          CDF=DEXP(-YY)
278:          IF(ICDF.EQ.1) GO TO 10
279:          EY=EE+YY
280:          IF(EY.GT.200.) EY=200.
281:          PDF=ALPHA*DEXP(-EY)
282:          GO TO 10
283:        4 CX21=XDEV**2+1.D0
284:          YMEAN=DLOG(XMEAN)
285:          YDEV=DSQRT(DLOG(CX21))
286:          Z=(DLOG(X)-YMEAN)/YDEV
287:          CDF=CDFNOR(Z)
288:          IF(ICDF.EQ.1) GO TO 10
289:          EZ=-(Z**2)*.5D0
290:          IF(EZ.LE.-200.) EZ=-200.
291:          PDF=SPI2*DEXP(EZ)/(YDEV*X)
292:          GO TO 10
293:  5         TEMP=(BETA/X)**ALPHA
294:            CDF=DEXP(-TEMP)
295:            IF(ICDF.EQ.1) GO TO 10
296:            PDF=CDF*TEMP*ALPHA/X
297:       10 RETURN
298:          END
299:          DOUBLE PRECISION FUNCTION XINV (Z)
```

This defined the performance function

This subroutine determines the point at which the CDF is evaluated for the maximum impact variable

Evaluates the CDF

```
300:            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
301:           F(X,P1)=P1-CDFNOR(X)
302:          Y=Z
303:          IF(Z.GT.0.5D0) Y=1.D0-Z
304:          C0=2.515517D0
305:          C1=0.802853D0
306:          C2=0.010328D0
307:          D1=1.432788D0
308:          D2=0.189269D0
309:          D3=0.001308D0
310:            T=(-2.D0*DLOG(Y))**.5D0
311:          DNUM=C0+T*(C1+T*C2)
312:          DNOM=1.0D0+T*(D1+T*(D2+T*D3))
313:          X=T-(DNUM/DNOM)
314:          IF(Z.LT.0.5D0) X=-X
315:            A=X
316:            B=X+.001D0
317:            V=F(B,Z)
318:            U=F(A,Z)
319:            XX=B
320:  1         CONTINUE
321:            IF(DABS(XX-A).GE.1.D-10) THEN
322:            XX=B-V*(B-A)/(V-U)
323:            A=B
324:            B=XX
325:            U=V
326:            V=F(XX,Z)
327:            GO TO 1
328:            END IF
329:            XINV=XX
330:            RETURN
331:            END
```

The inverse normal using the secant method

```
332:            DOUBLE PRECISION FUNCTION CDFNOR(Z)
333:  C THIS FUNCTION COMPUTES THE NORMAL CDF.
334:            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
335:            COMMON /TWO/PI,SPI2,PI2
336:          DATA A/0.31938153D0/,B/-0.356563782D0/,C/1.781477937D0/,
337:         +D/-1.821255978D0/,E/1.330274429D0/
338:          EZ=-(Z**2)*.5D0
339:          CDFNOR=0.0D0
340:          IF(EZ.LE.-200.0D0) GO TO 1
341:          ZX=SPI2*DEXP(EZ)
342:            IF(DABS(Z).GT.6.D0) GO TO 2
343:          T=1.D0/(1.D0+(0.2316419D0*DABS(Z)))
344:          CDFNOR=ZX*T*(A+T*(B+T*(C+T*(D+T*E))))
345:            GO TO 1
346:  2         Z2=1.D0/(Z*Z)
347:            CDFNOR=ZX*(1.D0-Z2*(1.D0-3.D0*Z2*(1.D0-5.D0*Z2)))/DABS(Z)
348:       1  IF(Z.GT.0.0D0) CDFNOR=1.0D0-CDFNOR
349:          RETURN
350:          END
```

```
351:          SUBROUTINE STAT(U,K1,K2,XM,STD,XMED,COV)
352:  C
353:  C*   THIS SUBROUTINE CALCULATES THE STATISTICS (MEAN,STD DEV,MEDIAN,COV)
354:  C*   OF Y FUNCTION.
355:  C
356:            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
357:          DIMENSION U(K2)
358:          XK=K2-K1+1
359:          XM=0.
```

```
360:          DO 63 I=K1,K2
361:          XM=XM+U(I)
362:       63 CONTINUE
363:          XM=XM/XK
364:          STD=0.
365:          DO 64 I=K1,K2
366:          STD=STD+(U(I)-XM)**2
367:       64 CONTINUE
368:          STD=STD/(XK-1.D0)
369:            STD=DSQRT(STD)
370:            COV=STD/XM
371:            XMED=XM/DSQRT(1.D0+COV**2)
372:          RETURN
373:          END
```

```
374:          SUBROUTINE GENX(DIST,ALPHA,BETA,X,XA,XMEAN,XSTD,ISEED)
375:          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
376:          COMMON /TWO/PI,SPI2,PI2
377: C
378:          IDIST=INT(DIST+.1)
379:          AA=RAN(ISEED)
380:          GO TO (1,2,3,4,5), IDIST
381: 1        X=BETA*(-DLOG(AA))**(1.D0/ALPHA)
382:          XA=BETA*(-DLOG(1.D0-AA))**(1.D0/ALPHA)
383:          RETURN
384: 2        BB=RAN(ISEED)
385:          E=DSQRT(-2.D0*DLOG(AA))
386:          X=E*DCOS(PI2*BB)*XSTD+XMEAN
387:          XA=-X+2.D0*XMEAN
388:          RETURN
389: 3        X=BETA-DLOG(-DLOG(AA))/ALPHA
390:          XA=BETA-DLOG(-DLOG(1.D0-AA))/ALPHA
391:          RETURN
392: 4        BB=RAN(ISEED)
393:          SDX=DSQRT(DLOG(1.D0+XSTD**2))
394:          UX=DLOG(XMEAN)
395:          W=DSQRT(-2.D0*DLOG(AA))*DCOS(PI2*BB)*SDX+UX
396:          X=DEXP(W)
397:          XA=DEXP(-W+2.D0*UX)
398:          RETURN
399: 5        X=BETA*(-DLOG(AA))**(-1.D0/ALPHA)
400:          XA=BETA*(-DLOG(1.D0-AA))**(-1.D0/ALPHA)
401:          RETURN
402:          END
```

GENX obtains random samples from the distributions

RAN is library uniform random number generator for CYBER 175

```
403:          SUBROUTINE SECA1(COV,ISIGN,ALPHA)
404:          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
405: C        ISIGN = 1; WEIBULL DIST.
406: C              = 2; FRECHET DIST.
407:          F(X,COV)=-(1.D0+COV**2)*GAMMA(X)**2+GAMMA(2.*X)
408:          IF(ISIGN.EQ.1) X1=COV**(1.08)
409:          IF(ISIGN.EQ.2) X1=COV**(.677)/2.33
410:          IF(ISIGN.EQ.2.AND.X1.GT..49D0) X1=.48999999
411: 7        IF(ISIGN.EQ.1) F1=F(X1,COV)
412:          IF(ISIGN.EQ.2) F1=F(-X1,COV)
413:          IF(DABS(F1).LE.1.D-10) GO TO 1
414:          X2=X1+.01D0
415:          IF(ISIGN.EQ.1) F2=F(X2,COV)
416:          IF(ISIGN.EQ.2) F2=F(-X2,COV)
417:          XX=X2
418: 10       CONTINUE
419:          IF(DABS(XX-X1).GE.1.D-9) THEN
```

Secant method for computing Weibull and Frechet exponents

```
420:          XX=X2-F2*(X2-X1)/(F2-F1)
421:          X1=X2
422:          X2=XX
423:          F1=F2
424:          IF(ISIGN.EQ.1) F2=F(XX,COV)
425:          IF(ISIGN.EQ.2) F2=F(-XX,COV)
426:          GO TO 10
427:          END IF
428:          X1=XX
429:    1     ALPHA=1.D0/X1
430:          RETURN
431:          END
```

```
432:          SUBROUTINE WEI(XMEAN,XDEV,ALPHA,BETA)
433:          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
434:          COV=XDEV/XMEAN
435:          CALL SECA1(COV,1,ALPHA)
436:          AL1=1.D0/ALPHA
437:          BETA=XMEAN/GAMMA(AL1)
438:          RETURN
439:          END
```

Computes Weibull parameters

```
440:          SUBROUTINE FRE(XMEAN,XDEV,ALPHA,BETA)
441:          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
442:          COV=XDEV/XMEAN
443:          CALL SECA1(COV,2,ALPHA)
444:          AL1=1.D0/ALPHA
445:          BETA=XMEAN/GAMMA(-AL1)
446:          RETURN
447:          END
```

Computes Frechet parameters

```
448:          SUBROUTINE EVD(XMEAN,STD,ALPHA,BETA,PI)
449:          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
450:          ALPHA=PI/(STD*DSQRT(6.D0))
451:          BETA=XMEAN-.57721566490153/ALPHA
452:          RETURN
453:          END
```

Computes EVD parameters

```
454:          DOUBLE PRECISION FUNCTION GAMMA(Y1)
455:          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
456:          COMMON /TWO/PI,SPI2,PI2
457:          X=Y1+1.D+0
458:          Z=X
459:          IF(X.GE.6.0D+0)GO TO 456
460:          N=INT(X)
461:          Z=(6.0D+0)-N+X
462:    456   Y=1.D+0/Z**2
463:          ALG=(Z-.5D+0)*DLOG(Z)+.5D+0*DLOG(PI2)-
464:         $ Z-(1.D+0/(12.D+0*Z))*(((Y/0.14D+3-1.D+0/0.105D+3)*Y+
465:         $ 1.D+0/.3D+2)*Y-1.D+0)
466:          IF(X.GE.6.D+0)GO TO 457
467:          ITE=6-N
468:          DO 3 J=1,ITE
469:          A=X+J-1.D+0
470:          ALG=ALG-DLOG(A)
471:     3    CONTINUE
472:    457   GAMMA=DEXP(ALG)
473:          RETURN
474:          END
475:
```

The gamma function

*

Note: The performance function must be introduced in subroutine LSFMC.

For an example of subroutine LSFMC, see the last page of Appendix B.

References:

1. Ang, A. H. S. and Tang, W., Probability Concepts in Engineering Planning and Design, Vol. II, Wiley, 1984.

2. Box, G. E. P. and Muller, M. E., "A Note on the Generation of Random Normal Deviates," Annals of Math. Stat., 29, 1958, pp. 610-611.

3. Gumbel, E. J., Statistics of Extremes, Columbia U. Press, 1958

4. Hahn, G. J. and Shapiro, S. S., Statistical Models in Engineering, Wiley, 1968.

5. Hines, W. H. and Montgomery, D. C., Probability, Statistics, in Engineering and Management Science, 2nd Ed., Wiley, 1972.

6. Mann, N. R. et al., Methods for Statistical Analysis of Reliability and Life Data, Wiley, 1974.

7. Wirth, N., Algorithms + Data Structures = Programs, Prentice-Hall, 1976.

APPENDIX F

A New Algorithm for Estimating the Probability Distributions of Complicated Structural Response Functions

Y.-T. Wu

Southwest Research Institute

Introduction

The structural reliability analysis methods developed during the past ten to fifteen years can be used to establish the CDF of complicated structural response function by forming the so-called limit state function or performance function [1]. In the application of these methods [2 - 4], the Taylor's series expansions are taken at the most probable points. For a given response function value, there is a corresponding most probable point which needs to be found using proper optimization or iteration algorithm. Because at each of the most probable point, there is no error in the function and the error is small around the most significant region for probability calculations, reasonably accurate solutions are assured. Indeed, experience has indicated that the applications of these methods usually results in high quality CDF estimation. However, when the response function is complicated, and the computations of the response variables are tedious, the above methods tend to be difficult to be implemented and/or are prohibitively time-consuming.

Presented here is a more efficient scheme which is suitable for constructing the cumulative distribution function (CDF) of any complicated function which has no explicit form, i.e., the objective function can not be expressed in algebraic form. The method is particularly suitable for the cases where the computation of the objective function is time consuming such that the Monte Carlo method becomes prohibitively costly.

Efficient Method of Constructing CDF using the Most Probable Points

The efficient method of constructing the CDF of a function starts with a linear approximation of the response function about the mean values of the independent random variables. Then the CDF values and the associated most probable points for several "predicted" response function values will be computed. For any selected CDF value, however, the "predicted" response function is not accurate if the response function is nonlinear, therefore, the corresponding response function value will be "corrected" by solving the actual values at the predicted most probable point.

In order to show how the method works, a simple example is established to detail the above procedure. The example is a cantilever beam. The random variables involved are the applied force, P, and the length, L, which are assumed to be normally distributed variables. The mean and standard deviation of P are (0.223, 0.019) lbs. and the mean and standard deviation of L are (20, 1) inches. The maximum stress, S, at the fixed end of the beam is:

$$S = 787LP \tag{1}$$

The mean value of S is approximately 3500 psi.

Define the reduced variables u_1 and u_2 as

$$u_1 = (P - 0.223)/0.019 \tag{2}$$

$$u_2 = (L - 20.)/1. \tag{3}$$

Thus

$$P = 0.223 + 0.019u_1 \tag{4}$$

$$L = 20 + u_2 \tag{5}$$

By substituting equations 5 and 6 into equation 1, the stress becomes

$$S = 3510 + 300u_1 + 175u_2 + 15u1u_2 \tag{6}$$

By assigning a value for S, an iso-stress curves can be plotted on a two dimensional u space as shown in Figure 1. Note that u_1 and u_2 are standardized normal variables (with zero mean and unity standard deviation) because P and L are normal variables. Therefore, on the u_1,u_2 coordinate system, the joint probability density function is rotationally symmetric. The most probable point for a given S is easily identified as the point on a iso-stress curve which is nearest to the origin.

Now we can start the approximation procedure by taking the first order

expansion of S about the mean. Usually, one will operate on the L, P space and then transform to u_1, u_2 space. In this example, since the transformations are linear, we can use Eq. 6 directly. The first order expansion about the mean values ($u_1 = 0$ and $u_2 = 0$) results in

$$S = 3510 + 300u_1 + 175u_2 \tag{7}$$

This equation is exact at $u_1 = u_2 = 0$ (where S = 3510) only, but can be used as an approximation for other S values.

Based on Eq. 7, S is also a normal variable with a mean of 3510 and a standard deviation of 347. It is obvious, however, that the accuracy of the CDF of S will depend on the truncated higher order terms. Traditionally, a low order expansion (such as eq. 7) is only used to estimate the mean and the standard deviation. The CDF cannot be accurately approximated.

For illustration purposes, only one CDF value will be considered. Let S_1 = 3510 psi and S_2 = 4500 psi where S_1 curve is linear and passes through the origin and S_2 is parallel to S_1 in the u space. S_2 may be called a "predicted" stress since eq. 7 is assumed to hold for all the stress values. The predicted S_2 curve has a most probable point which is a point nearest to the origin. Assuming that eq. 7 is accurate, the first order reliability analysis method gives the following probability estimate:

$$P(S > 4510 \text{ psi}) = \Phi(-\beta) \tag{8}$$

where β is the minimum distance. The approximation, however, is not accurate because the most probable point derived was based on the inaccurate S equation. In fact, by substituting the most probable point (derived by assuming S=4500 in Eq. 7) into Eq. 6, the exact value is S = 4660 psi. The iso-stress curves S_1, S_2 and the exact S = 4660 are shown in Fig. 1. The exact curve is nonlinear and passing through the predicted most probable point. Since the predicted and the corrected curve match closely around the most probable region for S > 4660 (note that in Fig. 1, the two minimum distances are approximately equal.), the figure suggests

the following approximation:

$$P(S \text{ Exact} > 4660) = P(S \text{ Predicted} > 4510) \tag{9}$$

Mathematical Formulations

The above approximation can be formulated as follows. Let

$$Z(X) = a_0 + \Sigma a_i X_i + E = Z_p + E \tag{10}$$

where $Z(X)$ is a function of the random variables, X. Z_p is a random variable representing the sum of the Taylor's first order terms and E represents the sum of the higher order term. The error term should actually be a random variable, but in the present method it will be approximated by a deterministic value. E is defined based on the most probable point, i.e.,

$$E = Z(\text{most probable point for } Z_p = z_p) - z_p \tag{11}$$

where the most probable point is defined as a set of values of X which maximize the joint probability density function of X subjected to the constraint that $Z_p(X) = z_p$. The most probable point can be found using the reliability analysis method [1].

Define the exact deterministic value of Z as z, then,

$$\begin{aligned} P(Z > z) &= P(Z_p + E > z) \\ &= P(Z_p > z - E) \\ &= P(Z_p > z_p) \end{aligned} \tag{12}$$

where z_p is the predicted Z value using Z_p. Equation 12 can be stated as : the probability of exceedance of Z_p at z_p is approximately equal to the

probability of exceedance of the exact Z at the value of z computed using the most probable point of $Z_p=z_p$. By replacing Z by S, z_p by 4510, and z by 4610, Eq. 12 becomes Eq. 9.

To construct the entire CDF, the above demonstrated procedure can be repeated for other probability levels. Note that there is no limitation on the number of random variables and that the random variables can be any distribution.

The first order Z_p seems to be able to provide good approximation solutions as demonstrated in the following example. Improvements can be made by including the second order terms in Z_p. Alternatively, one can perform additional first order Z_p analyses at the tail regions using the predicted most probable points.

Establishing CDF - Example

The above algorithm has been used successfully to establish a CDF of a problem. The problem is similar to the previous cantilever beam problem except that the thickness of the beam is also modelled as a random variable. The goal is to estimate the CDF of the maximum stress.

Figure 2 shows the resulting CDF based on the analytical solution of the stress. CDF curves are plotted on the normal probability paper (the CDF of a normally distributed variable will be a linear line on this paper); the Y coordinate uses u as the basic unit where u is a standardized normal variates.

Using the conventional first-order-mean-expansion, the resulting CDF, in Fig. 2, is nearly a straight line indicating that S is approximately normal. This is because the approximating function is linear and the random variables studied are normal or nearly normal.

By applying the new algorithm, ten most probable points corresponding to ten CDF values are computed, and used to compute ten additional

deterministic solutions. These "new" stress values are the corrected values for the "old" CDF values. Figure 2 shows that the corrected CDF curve is very close to the "exact" (based on one million Monte Carlo simulations) CDF curve except at the region where u < -3, as shown in Fig. 2.

In this example, the difference between Z_e and Z_p ranges from 0.2 % (for u = -0.5) of Z_e to 32 % (for u = -4.3) of Z_e, suggesting that the response function is significantly nonlinear. This is reflected by the fact that, in Fig. 2, the corrected CDF curve is significantly non-normal. Therefore, by using the new algorithm, it is possible to assess the results by comparing Z_e and Z_p. Improvement is necessary only when the difference is large.

To improve the accuracy at the tail regions, there are two possible ways : (1). Take two more expansions at the tail regions (e.g., at u = -2.5 and u = 2.5). (2). Use quadratic or incomplete quadratic approximation about the mean values. The first method may be more appropriate when the quadratic approximations are difficult to obtain although the latter may provide more accurate results for problems involving highly nonlinear functions.

The performance of the new algorithm has also been evaluated using Fig. 3. In this figure, the CDFs of the three mean-based approximations to the exact solution are constructed to compare with the exact CDF. The three approximations are: linear, "incomplete" quadratic (second order mixed-terms are neglected), and (complete) quadratic expansions about the mean values of the independent random variables. By comparing the results of Fig. 3 with those of Fig. 2, it can be concluded that the new algorithm with only first order expansion performs better than the conventional quadratic expansion. Due to the fact that the complete quadratic approximations are much more difficult to obtain than the first

order approximations, the new algorithm with only first order approximation seems to be very suitable for estimating the CDFs for complicated functions.

Summary

The performance of the new algorithm using the demonstrated example is excellent by noting that only a number of deterministic solutions, in additional to the first-order-mean-expansion, are required. The results suggest that the new procedure is efficient and can be used to provide good CDF estimations for engineering analysis problems.

References

1. Ang, A. H.-S., and Tang, W. H., "Probability Concepts in Engineering Planning and Design," Vol. II, Wiley, 1984.

2. Fiessler, B., Neumann H. J., and Rackwitz, R., "Quadratic Limit States in Structural Reliability," Journal of the Engineering Mechanics Division, ASCE, Vol. 105, Aug. 1979., pp. 661-676.

3. Tvedt, L., "Two Second Order Approximations to the Failure Probability," Det Norske Veritas (Norway) RDIV/20-004-83, 1983.

4. Wu, Y.-T., "Demonstration of a New, Fast Probability Integration Method for Reliability Analysis," Proceedings of Symposium on Probabilistic Structural Design and Analysis, ASME, 1985.

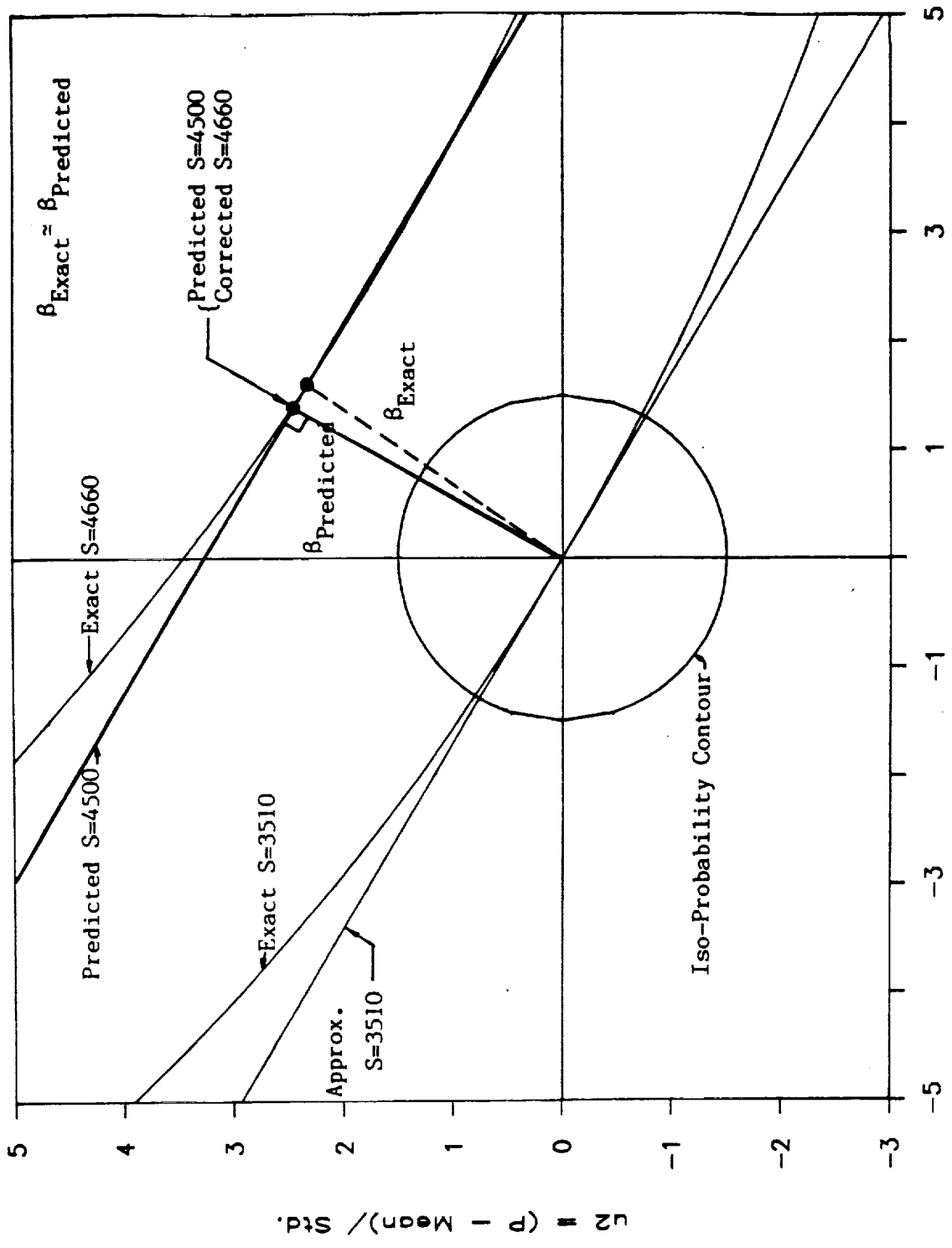
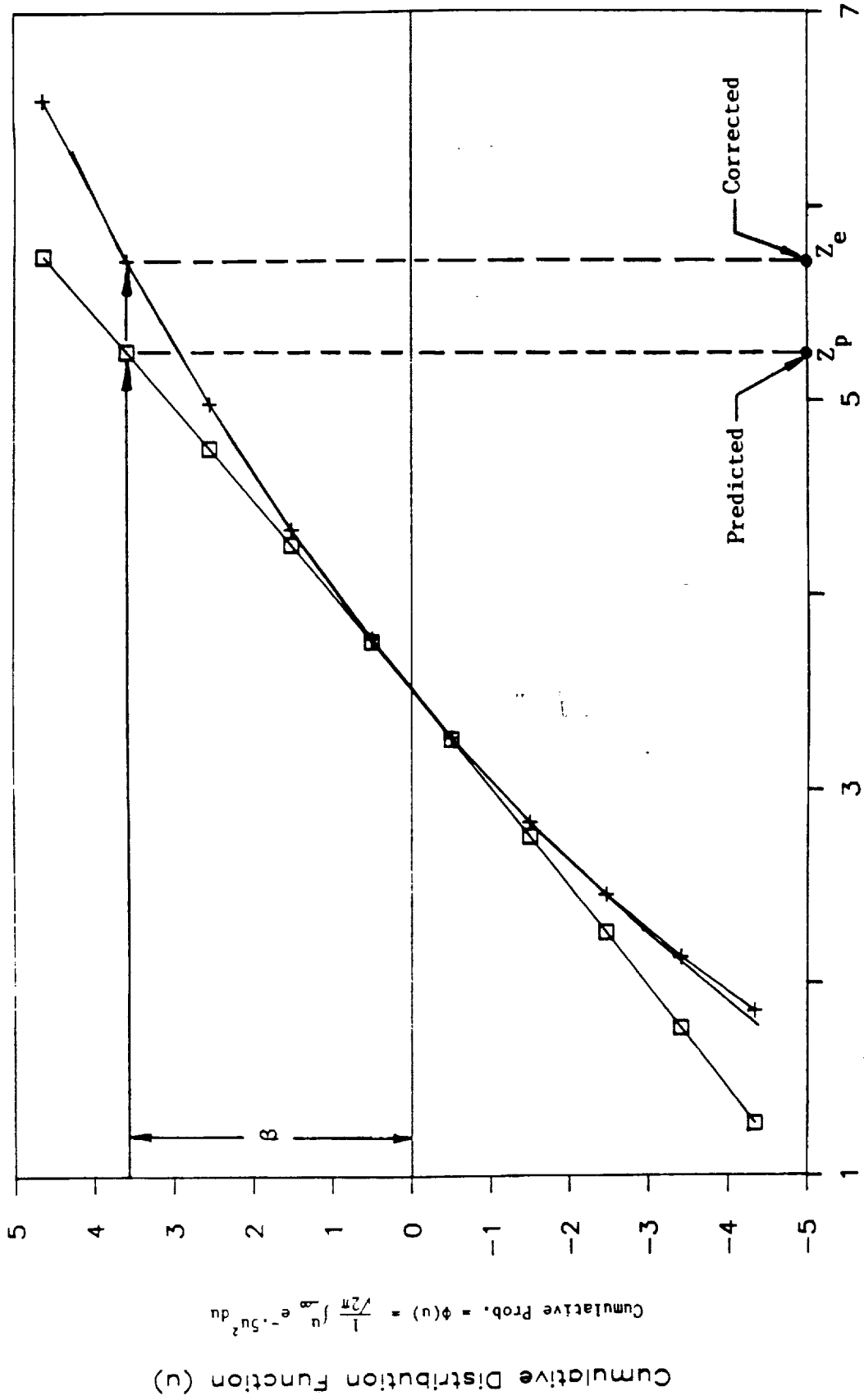


Figure 1. An Example showing the New CDF Estimation Algorithm

Figure 2. The Results of an Example using the New CDF Estimation Method



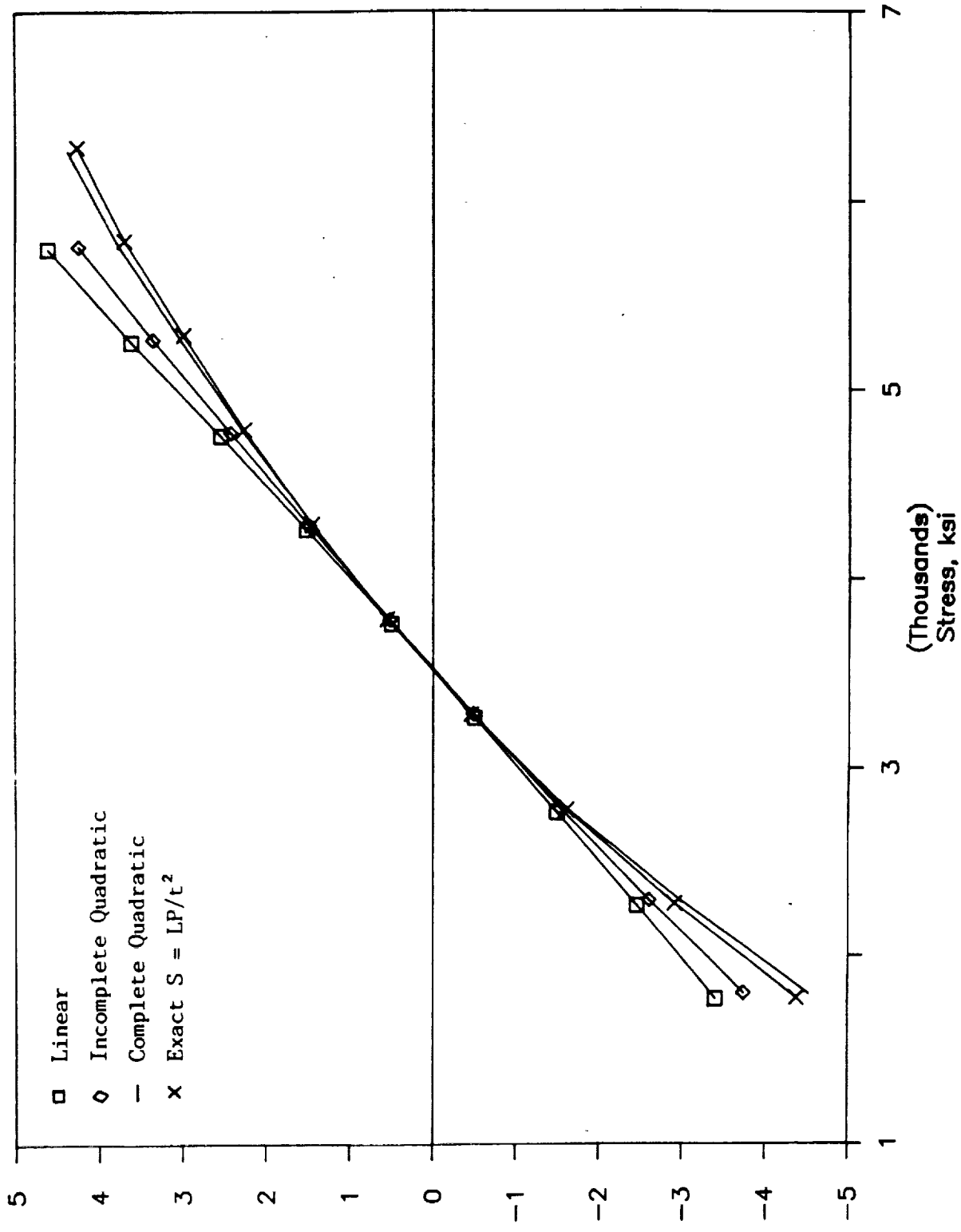


Figure 3. CDFs using Diffrent Approximation Levels

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1991	2nd Annual Contractor Report Nov. 86

4. TITLE AND SUBTITLE

Probabilistic Structures Analysis Methods for Select Space Propulsion System Components (PSAM)

5. FUNDING NUMBERS

WU-553-13-00
C-NAS3-24389

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Southwest Research Institute
San Antonio, Texas 78284

8. PERFORMING ORGANIZATION REPORT NUMBER

None

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR-187196

11. SUPPLEMENTARY NOTES

Project Manager, C.C. Chamis, Structures Division, NASA Lewis Research Center, (216) 433-3252.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 39

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

This second annual report summarizes the technical effort completed during the second year. The basic formulation for probabilistic finite element analysis is described and demonstrated on a few sample problems. This formulation is based on iterative perturbation that uses the factorized stiffness of the unperturbed system as the iteration preconditioner for obtaining the solution to the perturbed problem. This approach eliminates the need to compute, store and manipulate explicit partial derivatives of the element matrices and force vector, which not only reduces memory usage considerably, but also greatly simplifies the coding and validation tasks. All aspects for the proposed formulation were combined in a demonstration problem using a simplified model of a curved turbine blade discretized with 48 shell elements, and having random pressure and temperature fields with partial correlation, random uniform thickness, and random stiffness at the root.

14. SUBJECT TERMS

3-D elements; Hybrid-elements; Mixed methods initial stresses/strains; Random fields; Model output; Confidence levels; Temperature dependence; Validation cases; Verification; Monte Carlo; Integration algorithm

15. NUMBER OF PAGES

294

16. PRICE CODE

A13

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	